1992-12

Computer simulation studies of two-dimensional beamforming for linear arrays using a parallel computer system

Sullivan, Daniel Thomas

Monterey, California. Naval Postgraduate School

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Electrical and Computer Eng. Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): EC	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
Computer Simulation Studies Of Two-Dimensional Beamforming For Linear Arrays Using A Parallel Computer System (U)

12. PERSONAL AUTHOR(S)
Daniel Thomas Sullivan

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis		December 1992	77

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Governement.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Parallel Computers, Linear Arrays, Beamforming, Array Signal Processing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Computer simulation results of a parallel system conducting beamforming for a linear hydrophone array are presented. These studies were performed to determine the best mapping and partitioning of a sequential beamformer program onto a parallel system. Different partition designs and programming methodologies were examined, as well as latencies caused by inter-processor communications. Results of these simulation studies demonstrate that linear scalability in performance is possible by programming with host-node methodology and utilizing efficient inter-processor communications.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Chin-Hwa Lee	22b. TELEPHONE (Include Area Code): (408) 656-2190 / 22c. OFFICE SYMBOL: EC/Le

DD FORM 1473, JUN 86 Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited

COMPUTER SIMULATION STUDIES OF TWO-DIMENSIONAL BEAMFORMING FOR LINEAR ARRAYS USING A PARALLEL COMPUTER SYSTEM

by
Daniel Thomas Sullivan
Lieutenant, United States Navy
B.S.E.E., University Of Illinois, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1992

ABSTRACT

Computer simulation results of a parallel system conducting beamforming for a linear hydrophone array are presented. These studies were performed to determine the best mapping and partitioning of a sequential beamformer program onto a parallel system. Different partition designs and programming methodologies were examined, as well as latencies caused by inter-processor communications. Results of these simulation studies demonstrate that linear scalability in performance is possible by programming with host-node methodology and utilizing efficient inter-processor communications.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank Mr. Steve Howell of Naval Surface Weapons Center and Ms. Betts Wald of the Office Of Naval Technology for the Complex Systems Engineering/ Office Of Naval Technology (CSE/ONT) block funds that provided the hardware and software necessary to conduct this research.

This research could not have been completed without the guidance and support of my advisor, Professor Chin-Hwa Lee. Professor Lee would willingly take time from his very busy schedule to discuss problems I had with the work. He made learning interesting and fun. My appreciation also goes to Professor Lawrence Ziomek for his excellent unpublished acoustics book and for taking the time and effort necessary to be my second reader.

I am grateful to Professor Lee's research assistant Dan Zulaica for his help in learning the Express software.

Finally, I am most grateful for the love and support from my family. I thank my brother Douglas for his insights of digital signal processing and Fortran programming. I am thankful to my grandmother Mrs. Francis R. Fischer for instilling in me the values of hard work and integrity.

I. INTRODUCTION

This thesis investigates the application of linear array beamforming to a parallel computer system. Parallel processing here means execution of several processing tasks concurrently rather than one after the other (sequential or serial processing), as occurs in conventional computers. This thesis examines the use of general-purpose workstations working in cooperation. The objective is to speed up the beamforming operations.

Parallel processing allows several processors to achieve the computing power of a supercomputer. Parallel processing provides an attractive solution if an application requires a prohibitively high throughput or the required supercomputer has yet to be invented. Now parallel processing can provide a feasible and economical solution. A parallel computer system's overall performance is dependent upon the computing power of the individual processors and the latency of the inter-processor communication channels. Both computational and communication activities determine the performance of a parallel computer system.

Sequential programs are converted for use in a parallel system by mapping sections of the sequential program to the processors in the system. The programmer must consider the computational requirements of a section of a program and consider what communications are required to send data to and from the processor executing the code. Performance improvement of a parallel program is accomplished by repartitioning the sequential program onto the parallel system until satisfactory performance is achieved.

Automatic parallelization of sequential software does not exist. Therefore, writing parallel programs is more challenging than writing sequential programs. Due to explicit handling of communications, programming parallel computers is still more difficult than programming a sequential computer. [Ref. 1:p. 134]

This thesis maps a frequency-domain beamformer algorithm onto a parallel computer system. The input signal was modeled as a broadband sound source. Several partitioning

methods were implemented with varying results. In this parallel system, the Ethernet provided inter-processor communications. As a result, the Ethernet added latencies to the total execution time. Ethernet delays not only affected inter-processor communications but also idle times. For fair comparison in this study, identical input data was used for all simulations of different partition methods.

Chapter II builds the foundation of linear array beamforming. Beamforming serves as the computational load for the processors. General three-dimensional theory is presented and the linear array equations are derived. Chapter III describes parallel computer systems, and the advantages they offer over sequential computers. Two distinct parallel programming methodologies are used in this study: Loosely Synchronous and Host-Node. Chapter IV describes the Loosely Synchronous experiments. Loosely Synchronous programming offers ease and flexibility at the expense of performance.

In Chapter V, the partitioning is done according to a Host-Node methodology. The performance is improved over the previous methodology; however, it is apparent that carefully chosen partitions are needed to yield better scalability. Chapter VI presents a more realistic beamformer partition which did improve overall performance. The partition in Chapter VI also demonstrated how inter-processor communications can increase communication and idle time. Inter-processor communication can degrade overall system performance if it is not handled properly.

Based on the results obtained in Chapter VI, Chapter VII presents a host-node partition that uses an efficient broadcast mechanism to reduce communication and idle times. This method is successful in improving scalability for up to eight nodes. Chapter VIII presents an entirely new method of partitioning. Instead of having each node dedicated to beamsteering a sector of angles, now each node processes a separate frequency spectrum for all angles. This improved overall performance since less data needed to be transmitted in the network. Also, each node performs fewer floating point operations.

Chapter IX measures the FFT and inter-processor communication times. A quadratic model is developed to model and predict the system performance with larger FFT computations. Finally, all partition results are compared in the conclusion.

II. THEORY OF TWO-DIMENSIONAL BEAMFORMING

In this chapter the theoretical background of beamforming is introduced. The purpose is to describe the tasks carried out in the parallel computer system.

A. LINEAR ARRAY FUNDAMENTALS

A signal received at a linear sensor array is usually contaminated by noise in amplitude and phase. In the ocean, in addition to the signal generated by the target, there are signals present due to reverberations from the ocean surface or sea layers. Worst yet, signals from multiple sources may combine in a coherent or incoherent manner. Signals can undergo reflection causing the received source signal to be amplitude and phase modulated.

The source produces a target signal that is transmitted through the ocean. The target signal is combined with self noise, such as vibrating machinery, slamming hatches, and other sounds. These self noises are random in nature. In addition to the source noise, each hydrophone element has its own sensor noise. The target self noise and hydrophone element noise can be modeled (at least over short times) as Gaussian stationary processes. The additive noise components are uncorrelated with the target signal. [Ref. 2:p. 8]

All underwater source signals have an amplitude and phase. In a physical system, the signals are real. The representation of a source signal with a phase modulated carrier and constant random phase θ is given by

$$u_r(t) = \sqrt{P}\cos(\omega_o t + \phi(t) + \theta) \tag{2.1}$$

The complex representation of Eq. 2.1 is

$$u(t) = \sqrt{P}e^{j(\omega_o t + \phi(t) + \theta)} \tag{2.2}$$

The phasor part of Eq. 2.2 is

$$\tilde{u}(t) = \sqrt{P}e^{j(\phi(t)+\theta)} \tag{2.3}$$

Equation 2.3 is the baseband complex envelope of the real signal $u_r(t)$. It represents the information carrying part of the source signal. For simplicity, the carrier frequency ω_0 of the source signal can be neglected in this model. The source signal is usually modeled as originating in the far-field of the array. It is assumed that the signal propagates through an isospeed medium. Since the medium introduces propagation delays, the signal at any element can be modeled as the time advanced or time delayed version of the signal received at the reference element.

Signals characterized by a single frequency are designated as narrowband signals. Signals with multiple frequencies are known as broadband sources. In this simulation, each target has a base frequency plus three harmonics. The power of the first harmonic is 6 dB less than the power of the base frequency. The second harmonic's power is 12 dB less than the base frequency power. Lastly, the third harmonic's power is 18 dB below the base frequency power. Consequently, each target in the model transmits a broadband source signal.

Figure 2.1 illustrates a target in the array's far-field, radiating a signal $u_0(t)$ that is propagating through an isospeed medium. The right element receives signal $u(t)$ which is a time delayed version of the target's signal $u_0(t)$. Let d be the distance between the right and left elements in the array. Further, let $v(t)$ be the signal received by the left element. The wavefront will arrive at the second sensor after propagating a distance of $d\cos\Psi$. Let c represent the propagation speed. The time delay between the left and right array element is τ, given by

$$\tau = \frac{d\cos\Psi}{c} \tag{2.4}$$

The signal received by the second sensor is

$$v(t) = u(t-\tau) \tag{2.5}$$

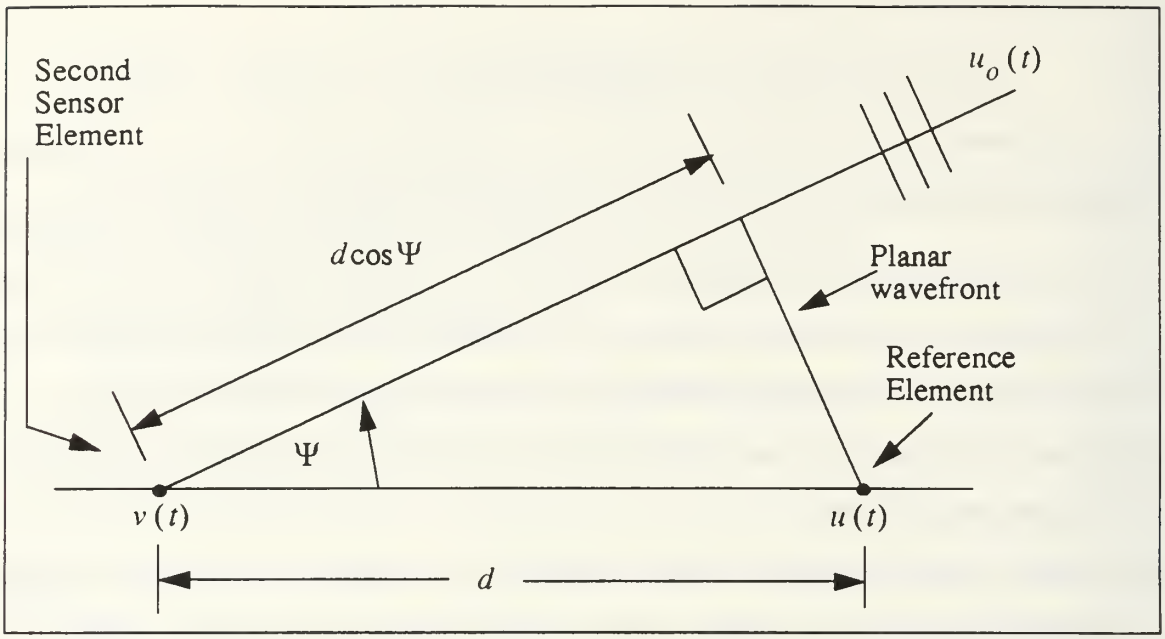

Figure 2.1 Linear hydrophone array.

If the carrier frequency ω_o is large compared to the bandwidth of the modulating signal $\phi(t)$, then the modulating signal can be considered quasi-static during time intervals on the order of τ. In this case, Eq. 2.2 reduces to

$$u(t) = \sqrt{P}e^{j(\omega_o t)} \tag{2.6}$$

Therefore,

$$v(t) = u(t)\,exp\left[\frac{-j\omega_o d\cos\Psi}{c}\right] \tag{2.7}$$

The time delay corresponds to a phase delay of the signal received by the reference element. The interelement spacing and the angle of arrival determine the phase delay. The phase delay is independent of time. [Ref. 2:pp. 11-12]

To avoid aliasing in the spatial domain, the interelement distance d must be equal to or less than half of the minimum wavelength of the source signal, that is,

$$d \leq \frac{\lambda_{MIN}}{2} \tag{2.8}$$

where λ_{MIN} is the minimum wavelength associated with the highest frequency component of the source spectrum.[Ref. 3:pp. 574-575]

B. GENERAL 3-D FORMULATION

In sonar applications, it is desired to classify a target by the frequency spectrum of its radiated field and to estimate the direction of arrival of the target signal. One way to accomplish this goal is to use the Fast Fourier Transform (FFT) algorithm to determine the spectral content, then maximize the array gain for each possible angle of arrival. The one-dimensional FFT approach is used in this simulation to allow for future adaptive beamsteering techniques. The following is a general three-dimensional formulation following the approach used in Ref. 3, pp. 626-628. The specific line array equations will be derived from the three-dimensional results.

Figure 2.2 shows a general plane wave traveling through an isospeed medium in the far-field of an array. The target signal is g(t). The target is located in the $\hat{n}_o$ direction. Therefore, its signal propagates in the $-\hat{n}_o$ direction where

$$\hat{n}_o = u_o\hat{x} + v_o\hat{y} + w_o\hat{z} \tag{2.9}$$

is a unit vector where

$$u_o = \sin\theta_o \cos\Psi_o , \tag{2.10}$$

$$v_o = \sin\theta_o \sin\Psi_o , \tag{2.11}$$

and

$$w_o = \cos\theta_o \tag{2.12}$$

are dimensionless direction cosines with respect to the X, Y, and Z axes. The position vector to a field point is given by $\underline{r}$, where

$$\underline{r} = x\hat{x} + y\hat{y} + z\hat{z} \tag{2.13}$$

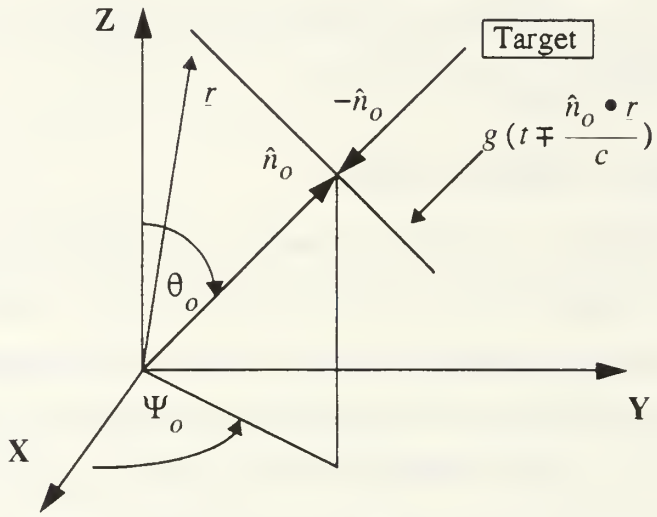

Figure 2.2 Three-dimensional plane wave case.

The array hydrophones are modeled as omnidirectional receivers having a linear gain. The acoustic field as a function of time and position is modeled by

$$y(t, \underline{r}) = g(t \mp \frac{\hat{n}_o \bullet \underline{r}}{c}) \tag{2.14}$$

To determine the spectral content of $y(t, \underline{r})$, the time-domain Fourier Transform is taken of Equation 2.14, that is,

$$Y(f, \underline{r}) = F_t\{y(t, \underline{r})\} \tag{2.15}$$

$$Y(f, \underline{r}) = F_t\left[g(t \mp \frac{\hat{n}_o \bullet \underline{r}}{c})\right] = G(f)\, exp\left[\mp j2\pi f(\frac{\hat{n}_o \bullet \underline{r}}{c})\right] \tag{2.16}$$

Substituting Equations 2.9 and 2.13 into Equation 2.16 gives the complex frequency spectrum of the target plane wave

$$Y(f, x, y, z) = G(f)\, exp(\mp j2\pi f_{xo}x)\, exp(\mp j2\pi f_{yo}y)\, exp(\mp j2\pi f_{zo}z) \qquad (2.17)$$

Variables f_{xo}, f_{yo}, and f_{zo} are the spatial frequencies in the X, Y, and Z directions in units of cycles per meter, where

$$f_{xo} = \frac{u_o}{\lambda} , \qquad (2.18)$$

$$f_{yo} = \frac{v_o}{\lambda} , \qquad (2.19)$$

and

$$f_{zo} = \frac{w_o}{\lambda} . \qquad (2.20)$$

Since the linear array of hydrophones is aligned along the X axis and if it is assumed that the target lies in the XY plane (θ_o=90°), then Equation 2.17 simplifies to

$$Y(f, x) = G(f)\, exp(\mp j2\pi f_{xo}x) \qquad (2.21)$$

$$Y(f, x) = G(f)\, exp(\mp j2\pi f \frac{\cos\Psi}{c} x) \qquad (2.22)$$

Equation 2.22 is the frequency-domain representation of the signal.

C. BEAMFORMING

To improve spatial resolution, the array output is amplitude weighted by Dolph-Chebyshev weights. Dolph-Chebyshev weighting offers the advantage of producing the narrowest beamwidth possible for a fixed ratio of main lobe to sidelobe level. The disadvantage of the Dolph-Chebyshev method is that there is a constant main lobe to side lobe ratio. There are other methods to improve spatial resolution. One physical means is to increase the array aperture. With a physical design of a linear array limited by cost, the

Dolph-Chebyshev amplitude weights provide an inexpensive method to improve spatial resolution. [Ref. 3:p. 558]

After the array outputs are amplitude weighted, the Fast Fourier Transform (FFT) is taken of the time samples from each element to determine the spectral content of the acoustic field. To find the angle of arrival of the target signal, linear phase weighting is applied to the FFT outputs from each sensor to steer the beam in a given direction. The array outputs are then summed. Since the noise is incoherent and uncorrelated with respect to the signal, the noise signals are canceled. The resulting sum is the frequency content of the acoustic field along a beamsteered direction.

If the spectral output of M elements are

$$Y_1(f, x_1) = G(f)\, exp\left(\mp j2\pi f \frac{\cos\Psi_o}{c} x_1\right) \tag{2.23}$$

$$Y_2(f, x_2) = G(f)\, exp\left(\mp j2\pi f \frac{\cos\Psi_o}{c} x_2\right) \tag{2.24}$$

$$\vdots \qquad\qquad \vdots$$

$$Y_M(f, x_M) = G(f)\, exp\left(\mp j2\pi f \frac{\cos\Psi_o}{c} x_M\right) , \tag{2.25}$$

then to steer the array to direction Ψ, the following linear phase weights (steering vector) would be needed:

$$W_1(f, x_1) = exp\left(\pm j2\pi f \frac{\cos\Psi}{c} x_1\right) \tag{2.26}$$

$$W_2(f, x_2) = exp\left(\pm j2\pi f \frac{\cos\Psi}{c} x_2\right) \tag{2.27}$$

$$\vdots \qquad\qquad \vdots$$

$$W_M(f, x_M) = exp\left(\pm j2\pi f \frac{\cos\Psi}{c} x_M\right) \tag{2.28}$$

where Ψ is the beam tilt angle. The array output for frequency f and angle Ψ is

$$Y_{out}(f, \Psi) = Y_1 W_1 + Y_2 W_2 + \ldots + Y_M W_M \tag{2.29}$$

In this simulation, the linear array detects targets with frequencies from zero to 512 Hz and with bearing angles from one to 180 degrees with one degree resolution. This hypothetical beamformer is used as a computational load for a parallel computer system. In the next chapter, a parallel processing system with message passing is introduced. Figure 2.3 illustrates the overall beamformer process.

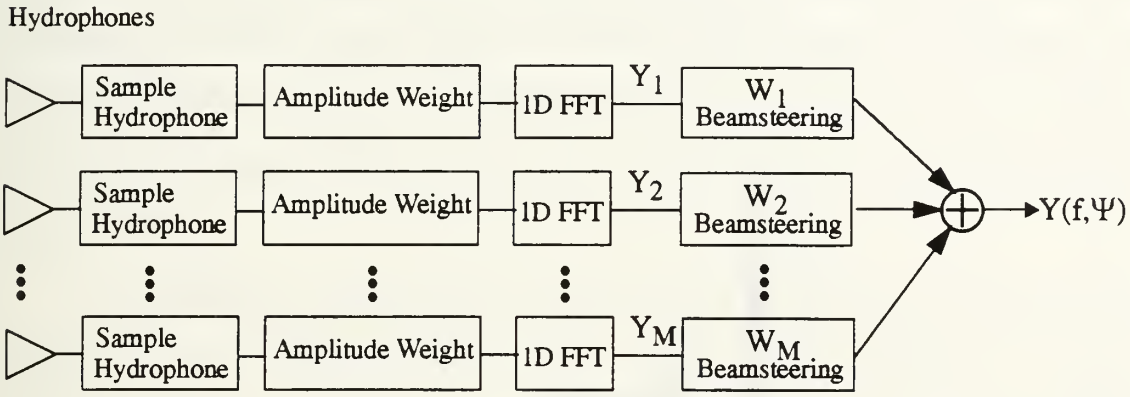

Figure 2.3 Overall beamforming diagram.

Figure 2.4 shows the beamformer output for one target source at bearing 90 degrees with a base frequency of 100 Hz and harmonics at 200 Hz, 300 Hz, and 400 Hz. The target's main source signal is seen as a peak at 90 degrees and 100 Hz.

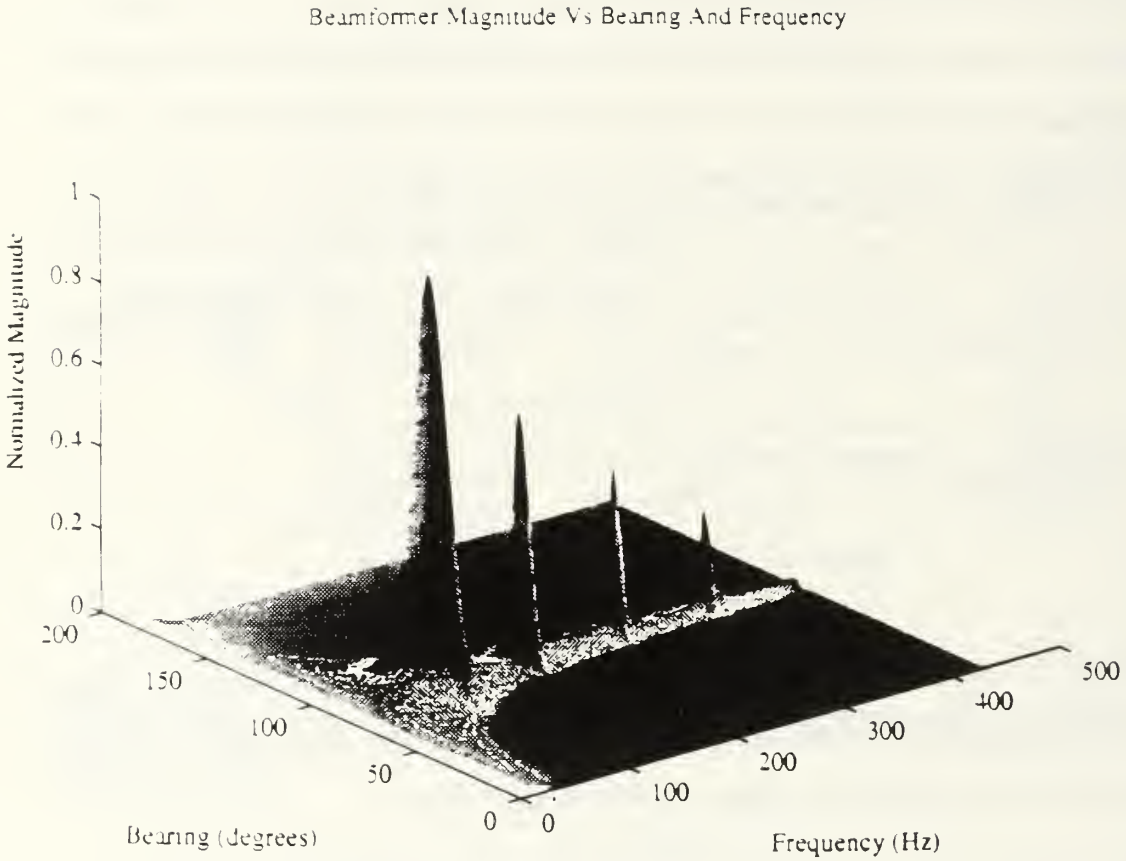

Figure 2.4 Beamformer output for one target source at broadside.

III. THEORY OF MESSAGE BASED PARALLEL COMPUTERS

A. MULTICOMPUTERS

A multiple instruction multiple data (MIMD) parallel computer consists of several independent computers connected in a network. Each of these independent computers acts as a node in the MIMD machine. Each node has its own private memory. Since each node can only access its own memory, global data must be transmitted via a communications network. As a result, MIMD systems are also called message-passing architectures. [Ref. 1:pp. 133-134]

A multicomputer's performance is affected by both the speed of the node computers and that of the communication network. The inherent characteristics of a numerical algorithm influences computation time. Communication speed is determined by inherent latencies and the network topology. Simply increasing the number of processors in a computer may not improve the speed because the node processors may be idle while waiting for the communication network to deliver data.

Compared to a second kind of system, shared memory machines, multicomputers with message passing offer increased scalability. The performance of shared memory machines is limited by bus saturation [Ref. 1:p. 116]. However, MIMD with message passing can improve computer power by increasing the number of processors. Linear scalability is an ideal goal. Scalability cannot be improved by only adding more processors to a multicomputer. To improve scalability, the programmer must partition the parallel program based on the processor computational power and communications bandwidth. [Ref. 1:p. 134]

Besides computation and communication time, network topology also influences the scalability. Early MIMD designers had to trade off the path lengths between nodes and the

number of physical connections at each node. Communications latency was proportional to the longest path between nodes in a certain network topology. [Ref. 1:p. 134]

B. PARALLEL PROGRAMMING ENVIRONMENT: EXPRESS

Express is a parallel programming environment used to create a multicomputer system using a network of independent processors. One processor is used as a host and the others as nodes. The Ethernet in a network of workstations provides communications between the processors. Express works in conjunction with the Unix operating system. While Unix is the operating system for each workstation, Express functions as a distributed operating system for the processors that together form a parallel system. The Express kernel provides the basic parallel computer functions for communications, sharing data, reading files, graphics, debugging, and performance analysis. Express performs these functions in a transparent manner. Express offers two basic programming methodologies: Host-Node and Loosely Synchronous [Ref. 4:p. 7]. In Loosely Synchronous methodology, only a single program is necessary to run all of the parallel system nodes. It is referred to as the Single Program Multiple Data system (SPMD). In the Host-Node methodology, each node of the network runs a different program.

1. Host-Node Programming Methodology

In the Host-Node Methodology, a host program runs on the host processor. One or more node programs run on the remaining node processors to implement the parallelization. Only the host program has access to input or output functions such as reading files and displaying data. All communications between the host and node processors are handled by message passing. Input data and computational results in the experiments are sent by two methods: Message Shuffle or Broadcast. [Ref. 4:p. 78]

a. Message Shuffle

The first method for inter-node and host-node communications is message shuffle. The host or node program transmits one message to a receiving node program in this scheme. Execution of the program at the receiver is blocked until the message is received. The message can be any data type. To ensure transmission integrity, the programmer must assign an identifier to the message. A node program waiting to receive a message will be blocked until a message arrives with the correct number of bytes and identifier. [Ref. 4:p. 89]

b. Broadcast

Besides message shuffle, a host or node program may communicate by broadcasting its message to several other nodes. The broadcast function blocks in the receiving node program until a message arrives with the correct number of bytes and identifier. The sender identifies the message recipients by defining a node list for a number of recipients. The node list is an array of node designators. [Ref. 5:p. 158]

2. Loosely Synchronous Programming Model

The Loosely Synchronous Programming Model was designed to make parallel programming easier. It is referred to as the SPMD methodology. The Host-Node Programming methodology requires separate host and node programs to execute on the processors. In the Loosely Synchronous Model (SPMD), the same program runs on all processors. One processor is designated as the host. The host acts as a file server; it serves node requests for operating system services. Loosely Synchronous programming offers the easiest transition of sequential programs to parallel programs. Loosely Synchronous programming reduces a significant amount of duplication of effort in writing parallel programs. Besides saving program development time, the Loosely Synchronous

Programming Model allows increased portability between different parallel computer architectures [Ref. 4:p. 26]. However, its scalability in performance is very poor.

In this thesis the objective is to explore several partition and mapping methods for beamforming to run on parallel systems. Each of the following chapters presents a partition method and its measured performance on the parallel workstations.

IV. LOOSELY SYNCHRONOUS SINGLE PROGRAM MULTIPLE DATA (SPMD) METHOD FOR PARTITIONING

In the Express environment the Loosely Synchronous Communication Method is called the Cubix Model of Programming. The Cubix Model of Programming is an output partition method. The host services all operating system requests from the nodes. The nodes execute the same program in a loosely synchronous manner. For these reasons, the Cubix Model of Programming is also referred to as the Single Program Multiple Data (SPMD) Method or Loosely Synchronous Communication Method for Partitioning [Ref. 6:p. 8]. Figure 4.1 illustrates the basic tasks of the beamformer program. For fair comparison among different partitioning simulations, all experiments were conducted with 96 hydrophone sensors. The target bearing was 90 degrees with a power of two watts. When the Cubix beamformer program begins execution, the nodes are in loosely synchronous mode. After parameter initialization, the host opens a file to store the beamformer results. Even though all nodes are running the same program synchronously, only the host node can access the operating system for services. After the data file is opened, the host program executes subroutine INDATA. Subroutine INDATA asks the user to specify the number of sources, the sensors, the direction, and the power of each source. If multiple sources are specified by the user, the user needs to input the amount of correlation between sources.

After the host program receives the input data from the user, the host program automatically sends the input data to all other nodes. Subroutine TRAN normalizes the correlation coefficients between sources. Next, the array elements are sampled. Subroutine CHEBWIN calculates the Dolph-Chebyshev amplitude weights for a main lobe to side lobe ratio of -80 dB. The array outputs are amplitude weighted and the one dimensional Fast Fourier Transform (FFT) is taken for the output of each array element.

Up to this point, the Loosely Synchronous beamformer program operates as a sequential beamformer. Parallelization of the beamformer operations occur in the next step where the nodes operate asynchronously. A sequential beamformer program would calculate for the sweep bearing angle Ψ from one degree to 180 degrees. The parallel computer system can save time by assigning each node a sector of sweep angles, and then writing the beamformer output data back to the host file in the proper sequence.

After each node calculates the spectral component at frequency f and angle Ψ, the beamformer output is written to the data file. In the Cubix Programming Model, the input/output is buffered. Data is stored in internal data structures. At program completion, the operating system flushes the data from the buffers in large packets to improve efficiency. Express allows the user to empty the data buffers with function KFLUSH [Ref. 5:p. 109]. When the Express library call KFLUSH is executed, all buffers are written to the host data file. Figure 4.2 illustrates how eight Sun workstations form a parallel computer system. One of the workstations functions as the Host and Node 0. The remaining workstations serve as Nodes 1 to 7. Ethernet provides the communications path between the host and nodes.

Express also provides a communications profiling utility called CTOOL. CTOOL assists developers because it details the amount of time each node has spent on computation, input/output, communications, system calls, and idle time. Table I details the profiling results of an eight-node Loosely Synchronous beamformer. In the eight-node Loosely Synchronized Communication Method, most of the time is used by input/output operations. The time spent by each node in each operation is illustrated in Figure 4.3.

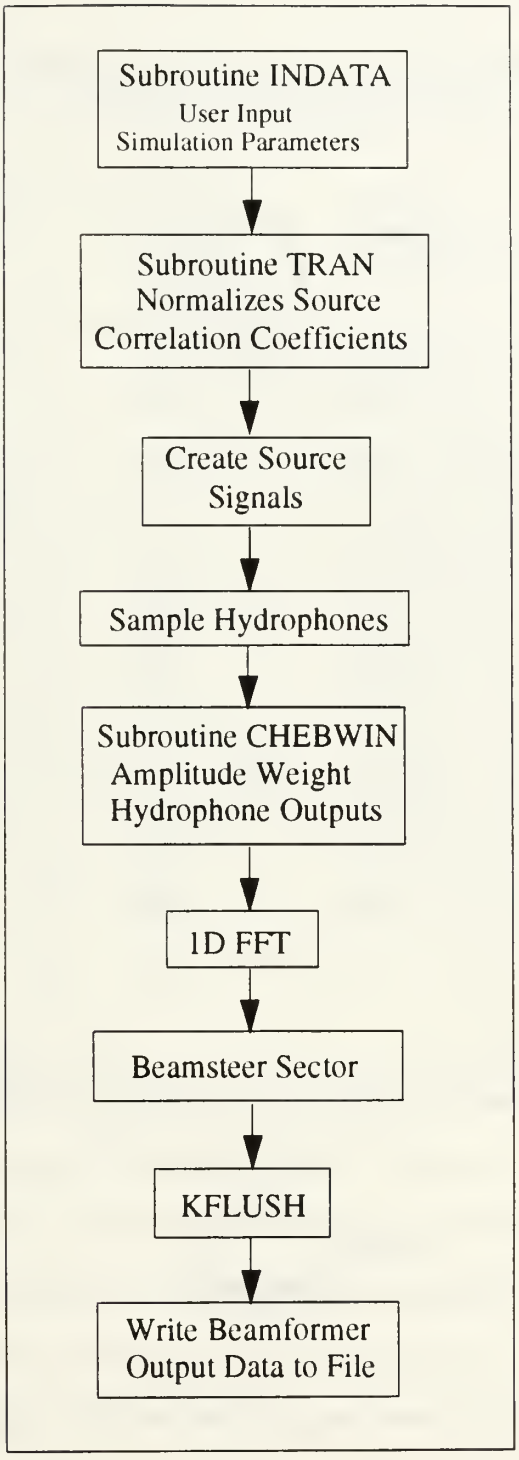

Figure 4.1 Loosely Synchronous beamformer program.

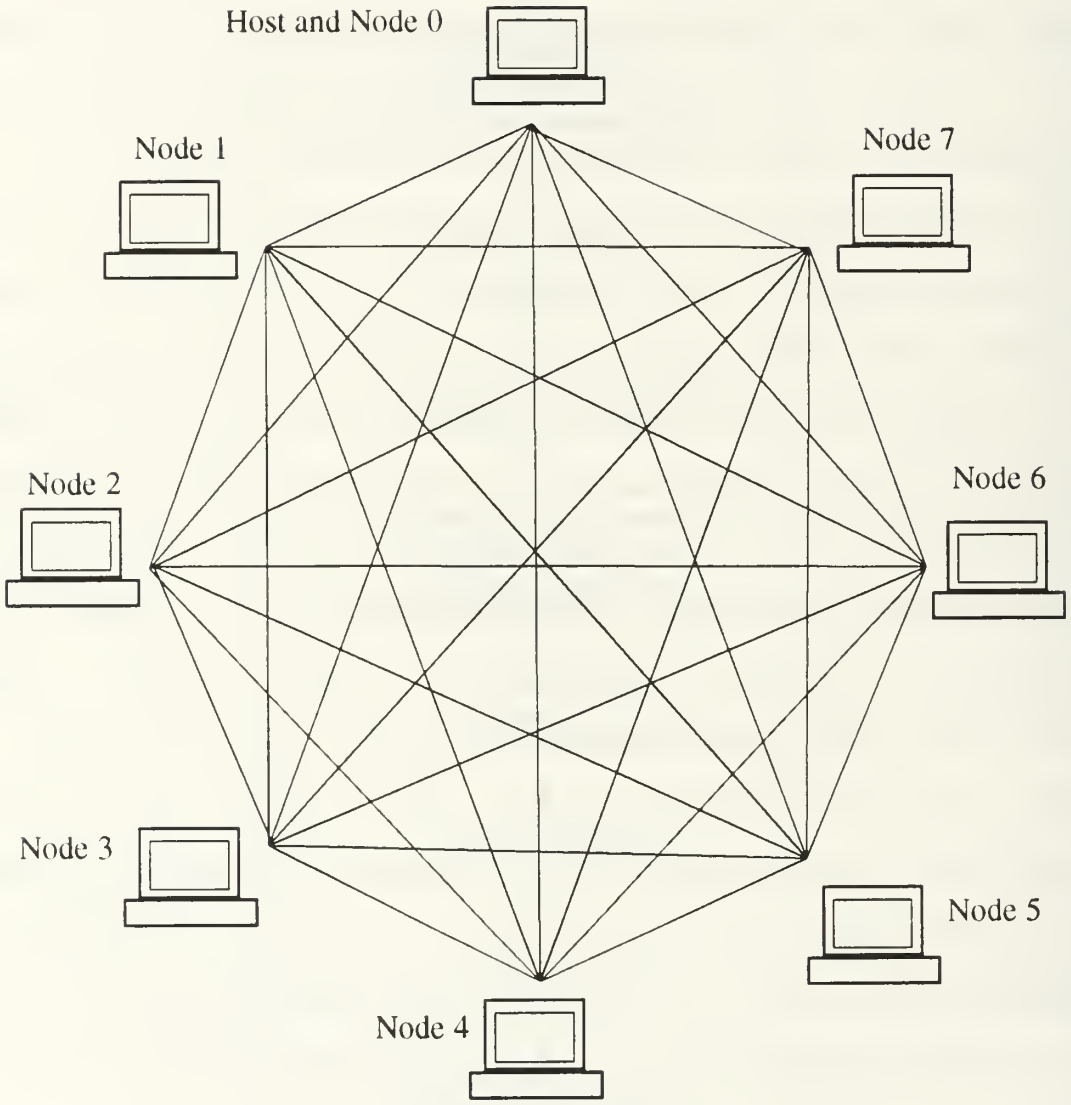

Figure 4.2 Fully connected eight node parallel computer system.

TABLE I: PROFILING DATA FOR LOOSELY SYNCHRONIZED COMMUNICATION METHOD OF PARTITIONING

Node	Calculation Time (msec)	Node Communication Time (msec)	Input/ Output (msec)	System Calls (msec)	Idle Time (msec)
Node 0	17434.66	0.00	42375.99	3616.57	191.00
Node 1	20509.46	0.00	40061.31	3005.16	50.33
Node 2	17913.60	0.00	42390.33	3151.93	168.57
Node 3	20563.21	0.00	41386.19	1639.65	47.03
Node 4	15370.41	0.00	44845.08	3302.68	94.20
Node 5	16217.64	0.00	44023.83	3325.08	52.88
Node 6	16862.01	0.00	43460.74	3055.21	80.41
Node 7	19231.14	0.00	41002.81	3175.71	37.60
Average	18012.77	0.00	42443.29	3033.99	90.26
Percent of Total	28.33%	0.00%	66.76%	4.77%	0.14%

In order to investigate the effect of incorporating more nodes in the parallel computer system, further experiments were conducted with a total of one, two, four, and six nodes used for each experiment. One measure of performance of a parallel system is speedup [Ref. 7:pp.8-9]. Speedup tells how much faster the beamformer program runs in a multi-node parallel computer than with only a host and a single node.

$$\text{Speedup} = \frac{\text{Processing time for entire task without using multiple processors}}{\text{Processing time for entire task using a single processor}} \quad (4.1)$$

Table II shows the average times of different activities for each experiment of beamformer programs with different total numbers of nodes in the system. Table III shows the total execution time and speedup when more nodes are used in the multi-node beamformer program.

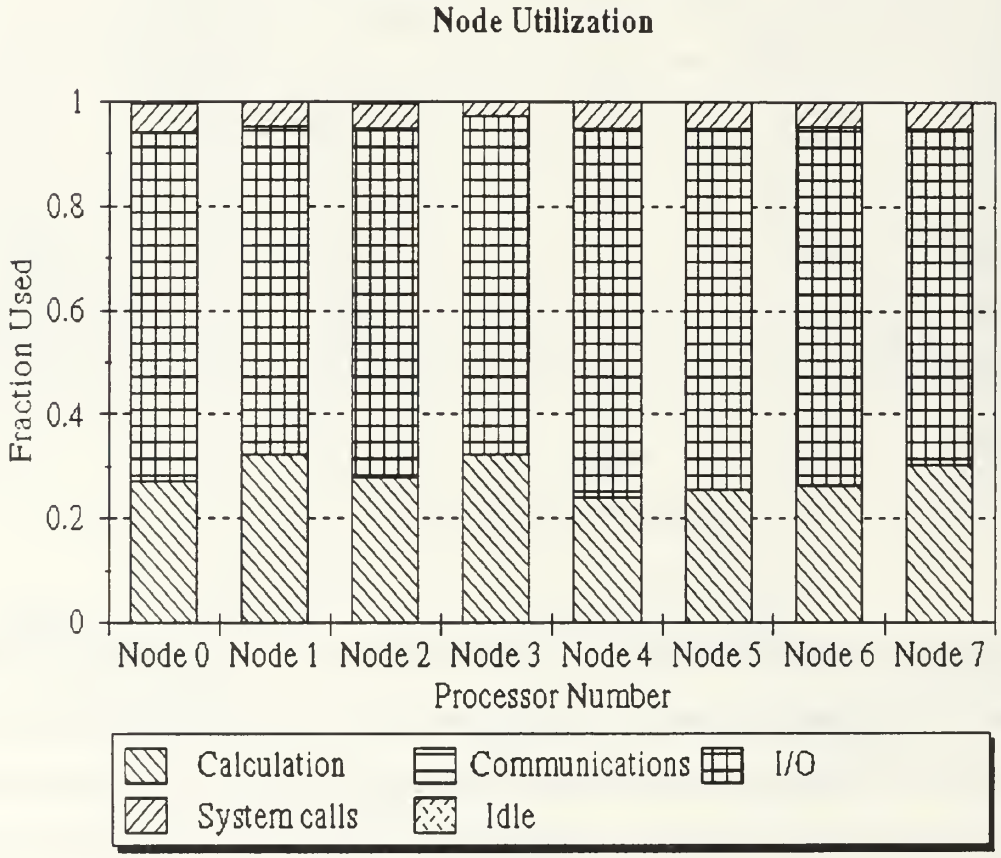

Figure 4.3 Node utilization for Loosely Synchronous Communication Method For Partitioning.

TABLE II: PROFILING DATA FOR LOOSELY SYNCHRONIZED COMMUNICATION METHOD FOR PARTITIONING FOR DIFFERENT NUMBER OF PROCESSORS

Number of Processors	Calculation Time (msec)/ Percent Of Total	Node Communication Time (msec)/ Percent Of Total	Input/ Output (msec)/ Percent Of Total	System Calls (msec) Percent Of Total	Idle Time (msec)/ Percent Of Total
One	87665.20 86.83%	0.00 0%	12954.15 12.83%	318.43 0.32%	25.52 0.02%
Two	47353.32 64.61%	0.00 0%	24340.42 33.21%	1507.96 2.06%	93.01 0.12%
Four	26008.40 46.86%	0.00 0%	27899.58 50.27%	1524.44 2.75%	65.20 0.12%
Six	19882.07 32.66%	0.00 0%	37816.80 62.13%	2969.35 4.88%	203.72 0.33%
Eight	18012.77 28.33%	0.00 0%	42443.30 66.76%	3034.00 4.77%	90.25 0.14%

TABLE III: TOTAL PROCESSOR TIME AND SPEEDUP FOR LOOSELY SYNCHRONIZED COMMUNICATION METHOD FOR PARTITIONING

Number of Processors	Average Total Processor Time (msec)	Speedup
One	100963.30	1.00
Two	73294.70	1.3775
Four	55497.60	1.82
Six	60871.90	1.66
Eight	63580.30	1.59

Figures 4.4 and 4.5 show plots of the data in Table III. As seen in Figure 4.4, total processing time decreased in the two and four processor programs compared to the single node beamformer. However, performance is degraded when the parallel beamforming is done on six and eight processors. This means that the performance deteriorates even if more nodes are used in the system. This is due to the increased input/output communication time used for each node.

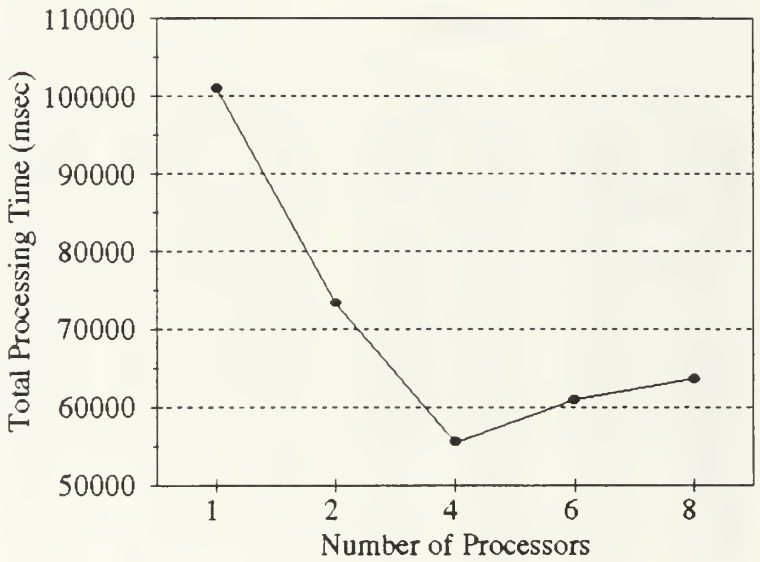

Figure 4.4 Total processing time versus number of processors.

The results shown here are not satisfactory. That is why other partition methods were explored; these are described in the next several chapters.

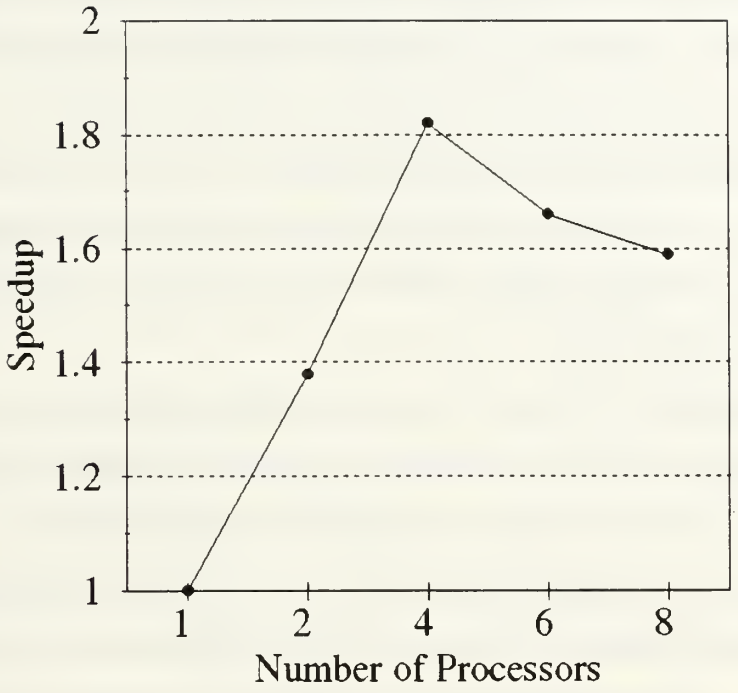

Figure 4.5 Speedup versus number of processors.

V. HOST-NODE PARTITION WITHOUT MESSAGE SHUFFLE

In Chapter IV it was demonstrated that the Loosely Synchronous Programming Model achieved decreased total processing time in the two and four node configurations. The six and eight node configurations experienced decreased performance. The Host-Node Programming methodology offers better performance because the user has greater flexibility in scheduling and partitioning the tasks. The Loosely Synchronous Programming Model allows for easier parallel programming but does so at the expense of flexibility. Partition Without Message Shuffle was the first partition method that used the Host-Node Programming methodology.

In this methodology, the host program is different from the node programs. Here, the program structure is similar to the Loosely Synchronous program. Each node executes its own program. In the Loosely Synchronous Programming Model, host and node communications were automatically facilitated via the Express programming environment. However, with the Host-Node methodology, the programmer must manually implement inter-processor communications. Express provides subroutines called KXWRIT and KXREAD for this purpose. Both are used for direct message passing between processors. Both host and node programs may use these system subroutines. KXWRIT sends a message to the specified recipient. A message can be any form of data, from a simple integer to a multidimensional array. The only limitation is that there must be sufficient buffer capacity in the sending and receiving processor to accommodate the message. The recipient processor receives the message by calling KXREAD. KXREAD will block the recipient's program execution until the message arrives. [Ref. 5:p. 204]

Figure 5.1 is a block diagram of the host and node program structure. The host program opens a data file and receives the user input data. If more than one acoustic source is specified by the user, the correlation coefficients between the sources are normalized. The host program sends the input data to each node. Each node program creates the source

signals, samples the hydrophones, and amplitude weights the array outputs. A one-dimensional Fast Fourier Transform (FFT) is performed on the output from each sensor element. Each processor handles a sector of bearing angles, which determines the beamformer output. A node program cannot access the Unix operating system to write the output to the data file. Each node program must use the Express point to point KXWRIT procedure to transmit the beamformer output to the host program. Once the host receives the beamformer data, the host program writes the results to the data file.

For fair comparison, the following set of input data was specified for all trials: one source at 90 degrees bearing with two watts of power, base frequency of 100 Hz, and an linear array consisting of 96 equally spaced sensors. Table IV shows the CTOOL profiling data of the trial for an eight processor parallel system with Partition Without Message Shuffle. Figure 5.2 illustrates the node utilization. Most of the processor time is consumed by performing calculations or waiting in an idle state. After a node program completes its calculations, it must wait to transmit the beamformer data back to the host. For the parallel programs using the Host-Node Programming methodology, the node input/output and system calls times are zero milliseconds. This is due to the Express restriction that only the host program has access to the Unix operating system [Ref. 4:p. 78].

Table V presents profiling results of the Host-Node Partition Without Message Shuffle in a parallel system of one, two, four, six, and eight processors. Table VI and Figure 5.4 display the speedup of each experiment. As the number of processors increases, the calculation time decreases. However, the idle time also increases. As a result, speedup is not a linear function of the number of processors. Figure 5.3 shows a plot of the system processing time. Compared to the two node system, processing time increases in the four node system because of the increase in idle time.

Partition Without Message Shuffle demonstrates that a parallel computer system using the Host-Node Programming methodology has the potential to achieve better scalability, i.e., a linear increase in speedup as the number of processors increase. A drawback of this partition method however, is that it is not realistic. If a linear hydrophone array were being

processed by a parallel system, each node most likely would not handle all of the sensors. Each node would process a group of sensors. Chapter VI presents a partition method that is a realistic system simulation.

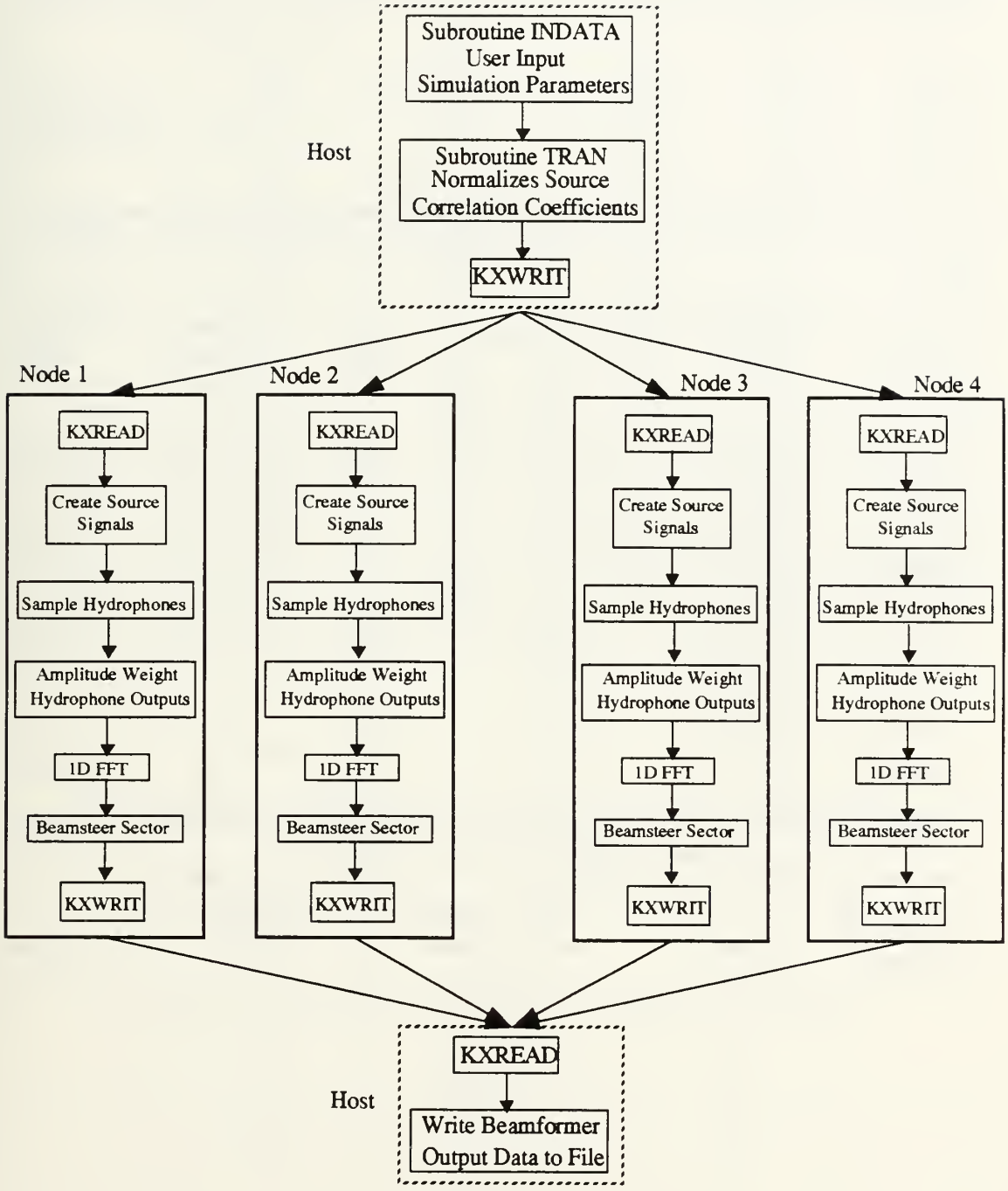

Figure 5.1 Partition Without Message Shuffle program.

TABLE IV: PROFILING DATA FOR EIGHT NODE PARTITION WITHOUT MESSAGE SHUFFLE

Node	Calculation Time (msec)	Node Communication Time (msec)	Input/ Output (msec)	System Calls (msec)	Idle Time (msec)
Node 0	17840.76	1007.50	0.00	0.00	18630.79
Node 1	17930.41	1029.03	0.00	0.00	18517.90
Node 2	36258.58	1205.04	0.00	0.00	23.76
Node 3	22702.70	1301.00	0.00	0.00	13454.23
Node 4	18078.65	1688.07	0.00	0.00	17712.95
Node 5	18560.72	1294.07	0.00	0.00	17599.93
Node 6	18903.14	1017.06	0.00	0.00	17558.41
Node 7	22185.56	989.45	0.00	0.00	14306.60
Average	21557.57	1191.40	0.00	0.00	14725.60
Percent of Total	57.53%	3.18%	0%	0%	39.29%

TABLE V: PROFILING DATA FOR PARTITION WITHOUT MESSAGE SHUFFLE

Number of Processors	Calculation Time (msec)/ Percent Of Total	Node Communication Time (msec)/ Percent Of Total	Input/ Output (msec)/ Percent Of Total	System Calls (msec)/ Percent Of Total	Idle Time (msec)/ Percent Of Total
One	110482.58 98.32%	1883.78 1.68%	0.00 0%	0.00 0%	12.55 0.012%
Two	56556.80 95.94%	1104.37 1.87%	0.00 0%	0.00 0%	1286.77 2.18%
Four	41167.08 62.27%	1244.63 1.88%	0.00 0%	0.00 0%	23703.06 35.85%
Six	27603.08 61.72%	612.78 1.37%	0.00 0%	0.00 0%	16509.16 36.91%
Eight	21557.57 57.53%	1191.40 3.18%	0.00 0%	0.00 0%	14725.60 39.29%

TABLE VI: TOTAL PROCESSING TIME AND SPEEDUP FOR PARTITION WITHOUT MESSAGE SHUFFLE

Number of Processors	Average Total Processing Time (msec)	Speedup
One	112375.58	1.00
Two	58947.93	1.91
Four	66114.78	1.70
Six	44725.02	2.51
Eight	37474.56	3.00

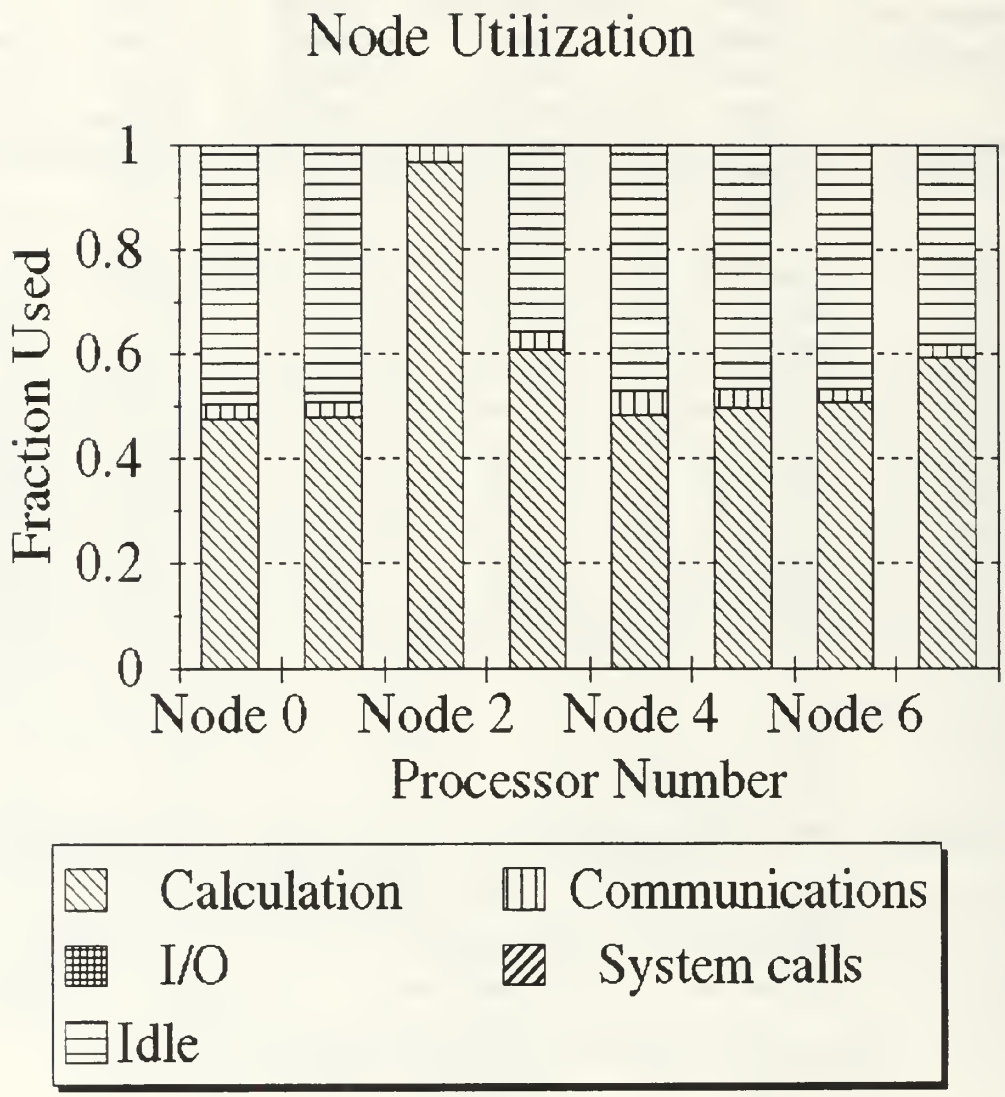

Figure 5.2 Node utilization for eight node Partition Without Message Shuffle method.

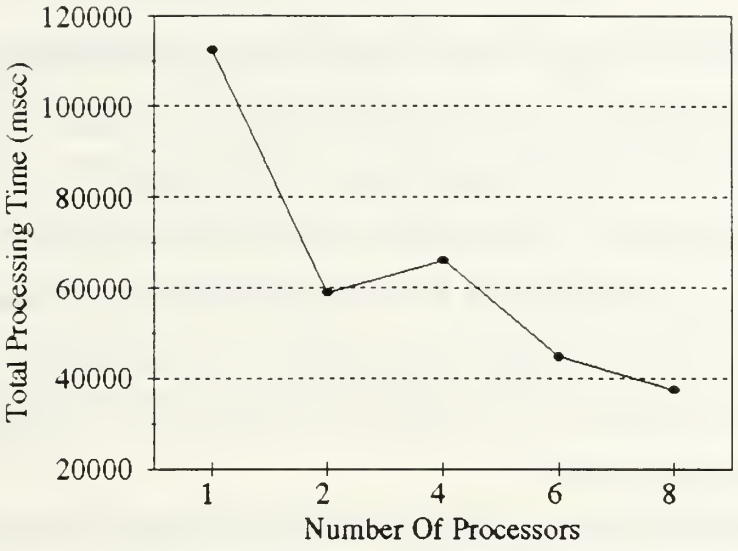

Figure 5.3 Total processing time versus number of processors.

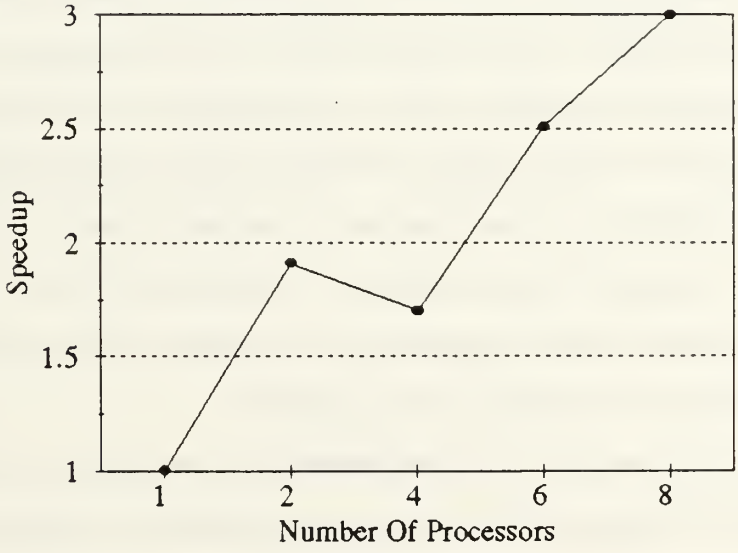

Figure 5.4 Speedup versus number of processors.

VI. HOST-NODE PARTITION WITH MESSAGE SHUFFLE

The major disadvantage of the Host-Node Partition Without Message Shuffle is that it is not a realistic system simulation. In an actual array system, each node processor would not process all of the hydrophone data. Instead, a node processor would most likely process only the data from a section of hydrophones. To steer the far-field beam pattern, each node needs to send its hydrophone data to the other nodes. Once all nodes have all of the hydrophone data, they can sweep their own sector of bearing angles and conduct the beamforming calculations. The Host-Node Partition With Message Shuffle is an implementation of these ideas.

Figure 6.1 illustrates a four processor parallel structure of the new Host-Node Partition With Message Shuffle. As before, subroutines INDATA and TRAN receive the user input and normalize the correlation coefficients between sources. As in the previous simulations, the array consists of 96 sensors and the target source is at 90 degrees bearing, radiating its signal with two watts of power. KXWRIT and KXREAD are used to transmit the message data from the host program to each node program. Each node program creates the source signal, then samples its respective group of hydrophones. For simplicity of illustration, Figure 6.1 shows four nodes in a parallel computer system. The total number of sensors in the array is 96. Therefore, to evenly balance the computing load, each node samples 24 sensors. Node 0 samples Group 0 (elements 1 through 24), Node 1 samples Group 1 (sensors 25 through 48), Node 2 samples Group 2 (elements 49 though 72), and Node 3 is responsible for Group 3 (hydrophones 73 though 96).

Each node program performs signal processing on the outputs from its hydrophone group. In order for a node program to steer the beam pattern, the node must have all of the hydrophone output data. Each node transmits its hydrophone data to the other nodes using the KXWRIT direct message-passing function. This is very time consuming since the node programs are blocked from continuing their execution until all data has been exchanged.

Once all hydrophone data is exchanged among the nodes, each node steers its far-field beam pattern according to its assigned sector of bearing angles. Finally, the beamformer output is sent to the host program to be written to a data file.

Table VII and Figure 6.2 show the node utilization for the Host-Node Partition With Message Shuffle of an eight node parallel system. In comparison to Partition Without Message Shuffle, Message Shuffle reduces idle time. Most of the time the nodes are performing calculations or communicating. Tables VIII and IX demonstrate that Partition With Message Shuffle had achieved decreasing processing time for two, four, and six processors. The eight processor run had less speedup than the six processor run. This is due to the increased communications between nodes in the eight node run which caused more processor idle time.

Figures 6.3 and 6.4 display the plots of the total processing time and the speedup using different number of processors in the runs. It is apparent that a new partitioning or communication scheme must be developed to improve the scalability beyond a six-processor parallel system.

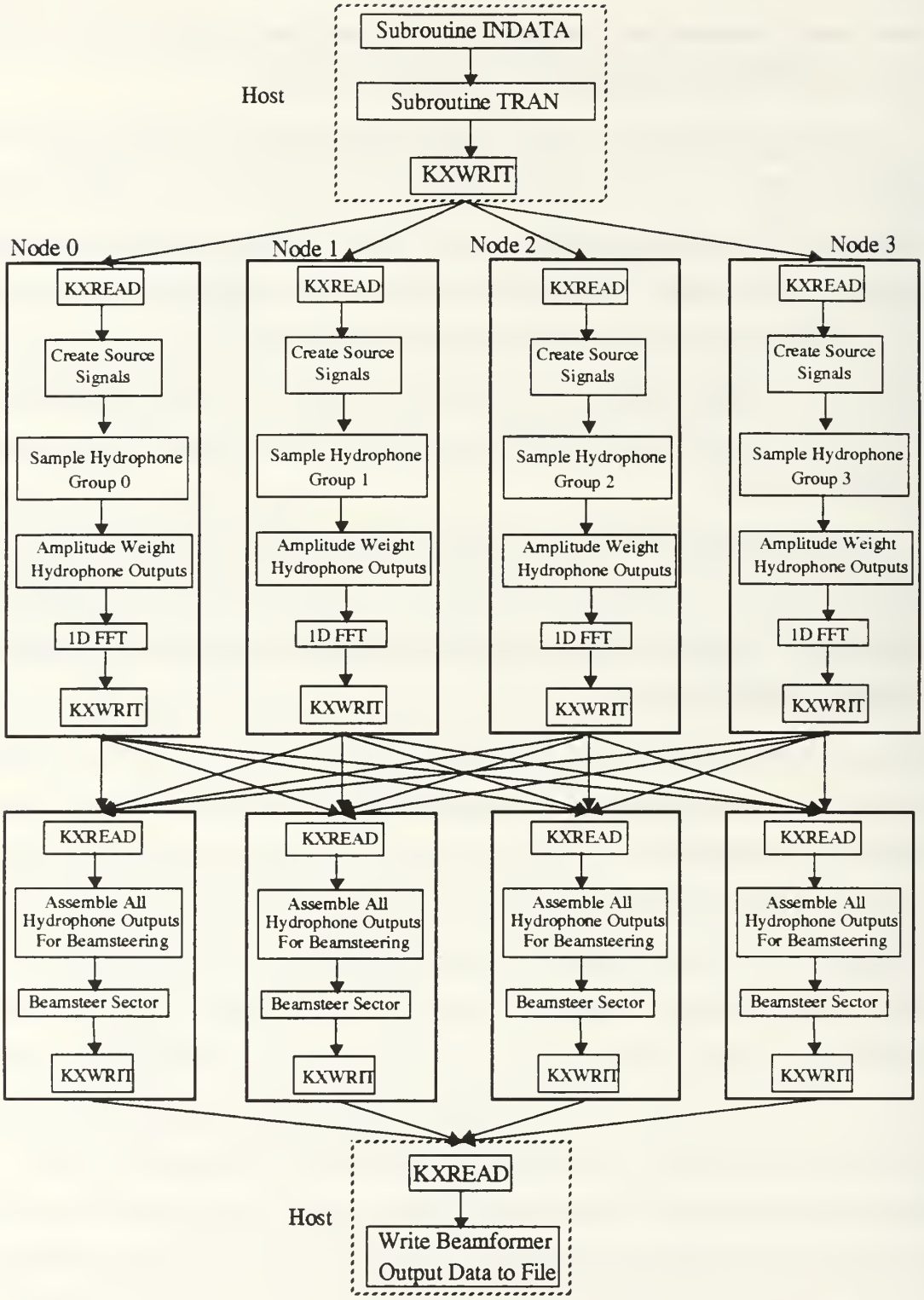

Figure 6.1 Four processor Partition With Message Shuffle program.

TABLE VII: PROFILING DATA FOR EIGHT NODE PARTITION WITH MESSAGE SHUFFLE

Node	Calculation Time (msec)	Node Communication Time (msec)	Input/ Output (msec)	System Calls (msec)	Idle Time (msec)
Node 0	20318.89	6087.46	0.00	0.00	7659.45
Node 1	17323.42	5649.29	0.00	0.00	11072.58
Node 2	17935.90	5754.61	0.00	0.00	10371.69
Node 3	22558.38	6080.17	0.00	0.00	5423.59
Node 4	24340.19	7054.94	0.00	0.00	2688.82
Node 5	19011.98	6777.81	0.00	0.00	8289.12
Node 6	28610.51	5452.54	0.00	0.00	9.69
Node 7	16140.80	7312.98	0.00	0.00	10599.94
Average	20318.89	6087.46	0.00	0.00	7659.45

TABLE VIII: PROFILING DATA FOR PARTITION WITH MESSAGE SHUFFLE

Number of Processors	Calculation Time (msec)/ Percent Of Total	Node Communication Time (msec)/ Percent Of Total	Input/ Output (msec)/ Percent Of Total	System Calls (msec)/ Percent Of Total	Idle Time (msec)/ Percent Of Total
One	110513.47 98.34%	1857.54 1.65%	0.00 0%	0.00 0%	9.85 0.01%
Two	58144.48 95.25%	2201.00 3.61%	0.00 0%	0.00 0%	695.25 1.14%
Four	30150.05 86.08%	2292.27 6.54%	0.00 0%	0.00 0%	2585.36 7.38%
Six	21547.69 66.77%	4533.28 14.05%	0.00 0%	0.00 0%	6189.18 19.18%
Eight	20318.89 59.65%	6087.46 17.87%	0.00 0%	0.00 0%	7659.45 22.48%

TABLE IX: TOTAL PROCESSING TIME AND SPEEDUP FOR PARTITION WITH MESSAGE SHUFFLE

Number of Processors	Average Total Processing Time (msec)	Speedup
One	112380.86	1.00
Two	61040.73	1.84
Four	35027.68	3.21
Six	32270.15	3.48
Eight	34065.79	3.30

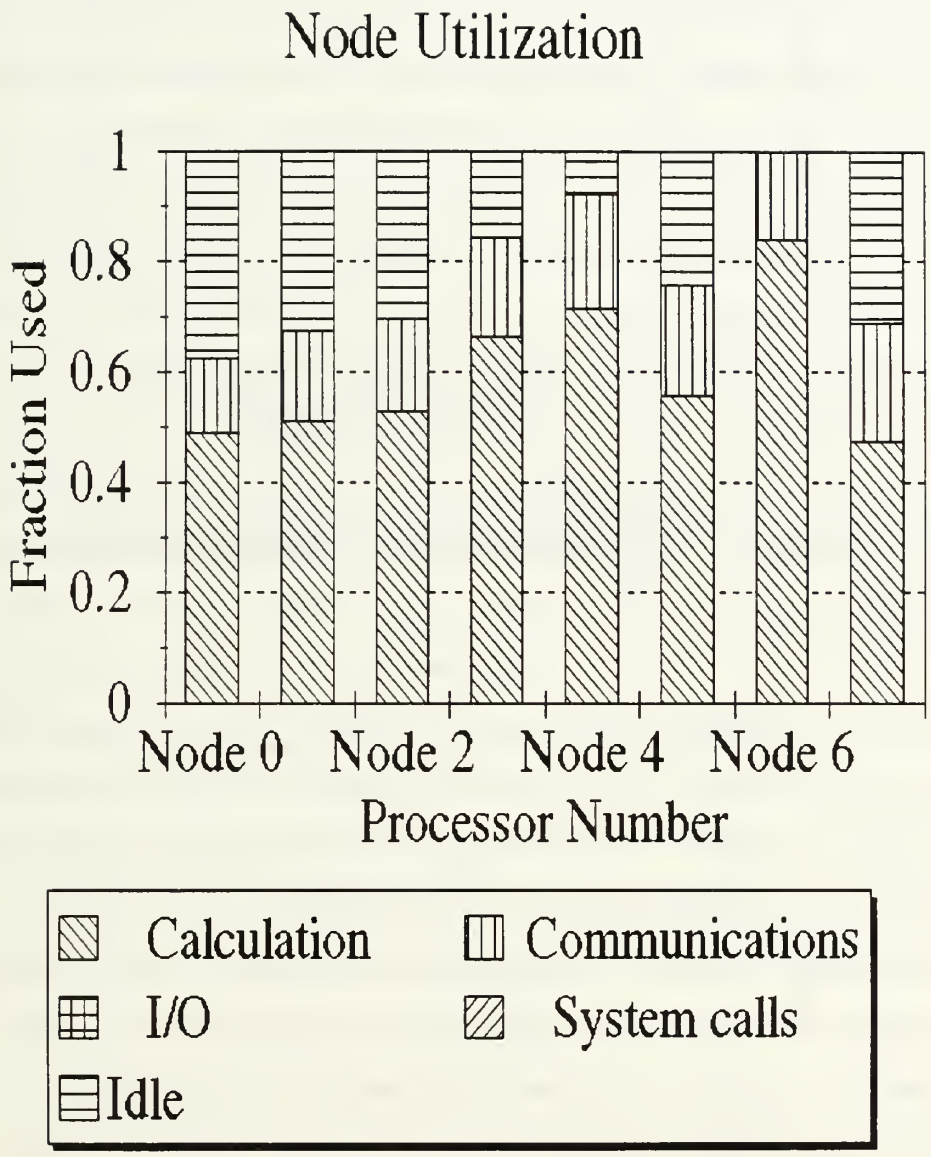

Figure 6.2 Node utilization for eight node Partition With Message Shuffle method.

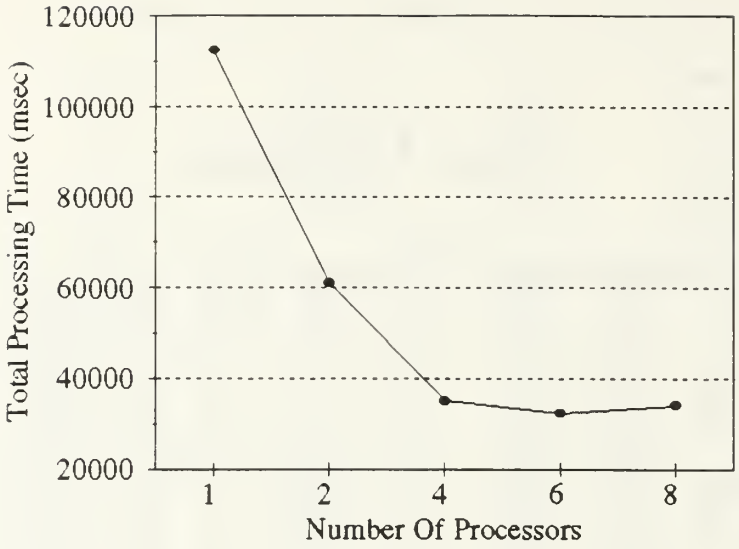

Figure 6.3 Total processing time versus number of processors.

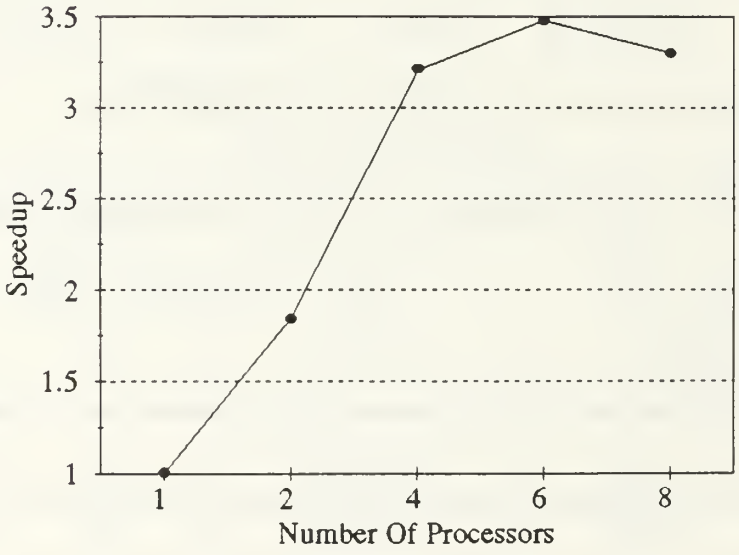

Figure 6.4 Speedup versus number of processors.

VII. HOST-NODE PARTITION WITH MESSAGE BROADCAST

The Host-Node Partition With Message Shuffle method performed satisfactorily in the two, four, and six-processor parallel system trials. Speedup dipped slightly for the eight-processor run. Host-Node Partition With Message Broadcast is a different method. Instead of using direct point-to-point message passing routines to send the hydrophone data between nodes, Partition With Message Broadcast uses a different communication procedure. In Message Broadcast, the Express subroutine program KXBROD is used. When a node is ready to send its hydrophone data, it broadcasts the data to all other nodes. All of the receiving node programs block execution until they receive the message. This message broadcasting paradigm significantly reduced processor idle time. Inter-node communication time remained almost identical compared to direct communication. The net result is that the Partition With Message Broadcast method improved speedup with all configurations of parallel systems considered here.

The Message Broadcast method simulations were conducted with the same inputs used for the previous partition methods. The host and node program structures are identical to Partition With Message Shuffle with the exception of inter-node communications. Figure 7.1 shows the call to KXBROD for each node to broadcast its hydrophone data to the other nodes. Table X and Figure 7.2 demonstrate that idle time is dramatically reduced using broadcast communications in an eight-node parallel system. The average idle time was 1514.37 milliseconds for Message Broadcast. The Partition With Message Shuffle method had an average idle time of 7659.45 milliseconds. The Partition With Message Broadcast method had only 19.77% of the idle time that the Partition With Message Shuffle method had in an eight-node parallel system.

Tables XI and XII display the system measurements as the number of processors is increased in a Partition With Message Broadcast. Figure 7.3 shows that total processing time remained almost constant when six to eight processors were used but it decreased

significantly when one, four, and six processors were used. Figure 7.4 illustrates that speedup consistently improved in the two, four, and six-processor cases. Speedup in the eight-processor system did not improve as much as it did in the six-processor system. The Host-Node Partition With Message Broadcast method was successful in demonstrating that an improvement in inter-node communications is essential for efficient multi-processor parallel systems.

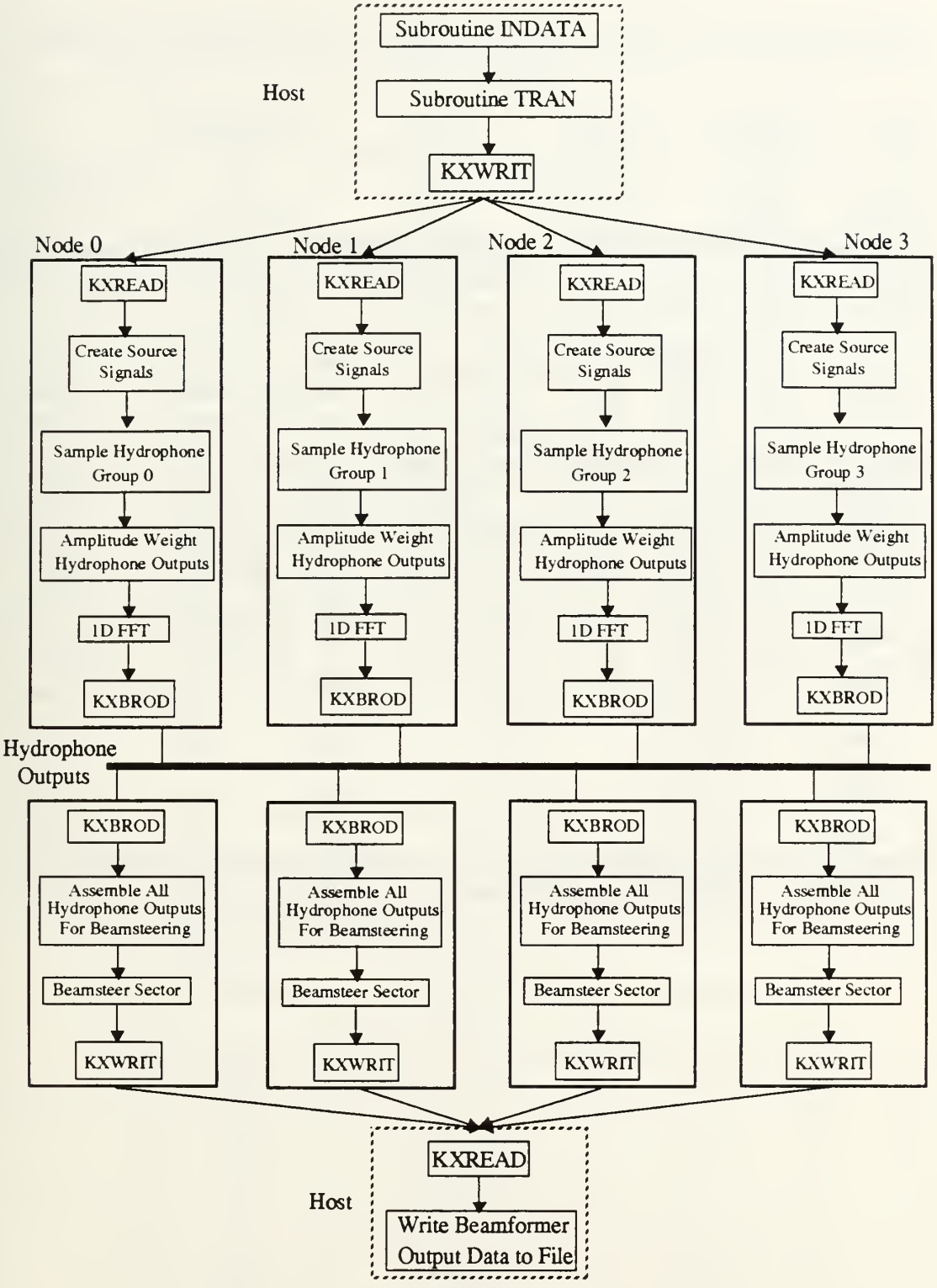

Figure 7.1 Four processor Partition With Message Broadcast program.

TABLE X: PROFILING DATA FOR EIGHT NODE PARTITION WITH MESSAGE BROADCAST

Node	Calculation Time (msec)	Node Communication Time (msec)	Input/ Output (msec)	System Calls (msec)	Idle Time (msec)
Node 0	18176.51	5171.22	0.00	0.00	1665.41
Node 1	17883.62	5285.72	0.00	0.00	1830.20
Node 2	18909.51	5290.17	0.00	0.00	801.34
Node 3	17650.70	5359.18	0.00	0.00	2016.23
Node 4	17935.77	5236.42	0.00	0.00	1856.73
Node 5	19515.40	5493.34	0.00	0.00	27.59
Node 6	17963.37	5326.50	0.00	0.00	1734.24
Node 7	25033.89	18098.78	0.00	0.00	2180.21
Average	18266.71	5239.68	0.00	0.00	1514.37

TABLE XI: PROFILING DATA FOR HOST-NODE PARTITION WITH MESSAGE BROADCAST

Number of Processors	Calculation Time (msec)/ Percent Of Total	Node Communication Time (msec)/ Percent Of Total	Input/ Output (msec)/ Percent Of Total	System Calls (msec)/ Percent Of Total	Idle Time (msec)/ Percent Of Total
One	110792.09 98.29%	1910.61 1.70	0.00 0%	0.00 0%	10.57 0.01%
Two	56569.40 94.08%	1578.61 2.63%	0.00 0%	0.00 0%	1980.55 3.29%
Four	31981.52 83.74%	3812.97 9.98%	0.00 0%	0.00 0%	2399.05 6.28%
Six	20256.19 78.79%	4168.62 16.22	0.00 0%	0.00 0%	1282.57 4.99%
Eight	18232.96 73.01%	5239.68 20.94%	0.00 0%	0.00 0%	1514.37 6.05%

TABLE XII: TOTAL PROCESSING TIME AND SPEEDUP FOR HOST-NODE PARTITION WITH MESSAGE BROADCAST

Number of Processors	Average Total Processing Time (msec)	Speedup
One	112714.26	1.00
Two	60128.55	1.87
Four	38193.54	2.95
Six	25707.38	4.38
Eight	25020.76	4.50

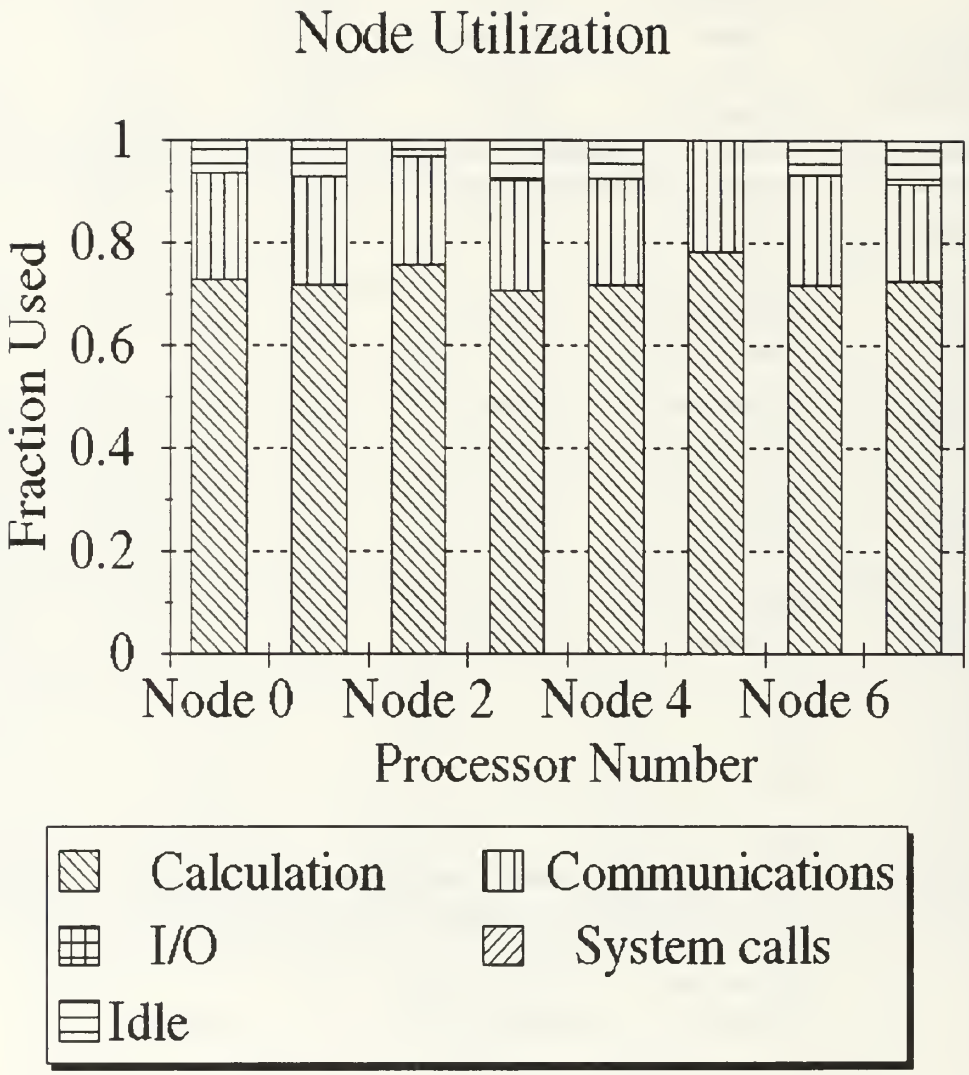

Figure 7.2 Node utilization for eight node Partition With Message Broadcast method.

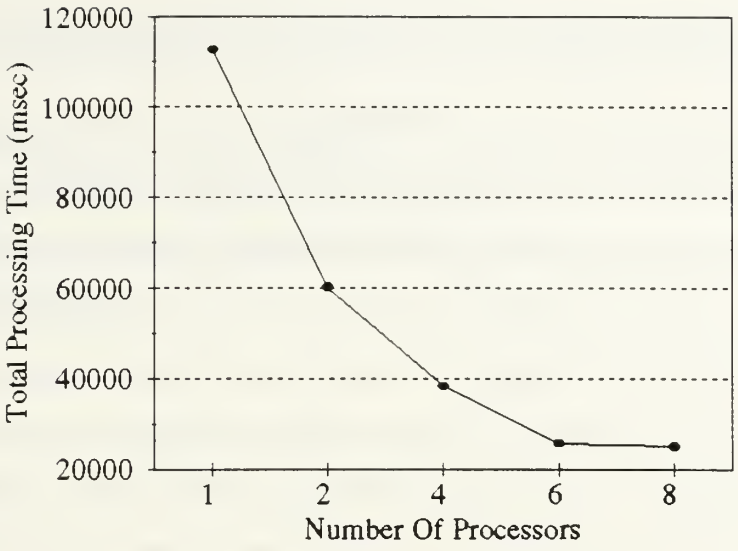

Figure 7.3 Total processing time versus number of processors.

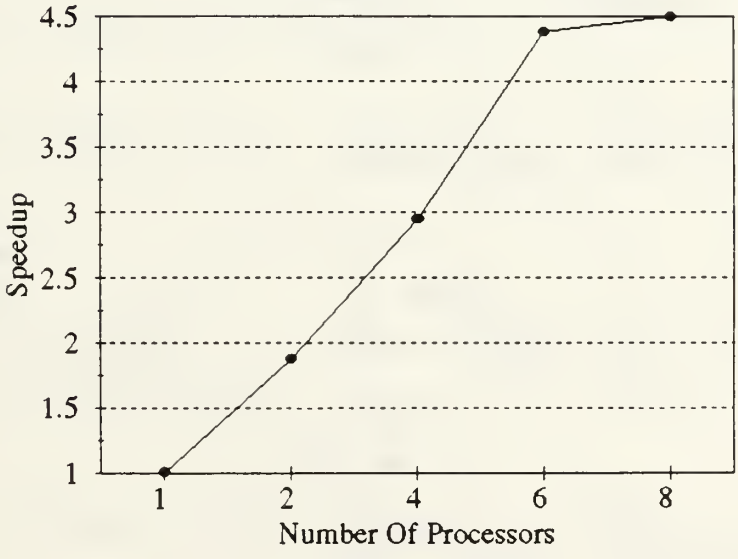

Figure 7.4 Speedup versus number of processors.

VIII. HOST-NODE FREQUENCY PARTITION WITH MESSAGE SHUFFLE

Host-Node Frequency Partition With Message Shuffle is a new partitioning method. It significantly reduces the inter-processor communications burden in a parallel computer. The net result is that each node has less communication load and therefore less idle time than in any of the previous partitioning schemes. For this reason, Frequency Partition With Message Shuffle had better speedup than any of the previous partitioning schemes.

Figure 8.1 is a diagram of the beamformer program structure. Consider the hydrophone outputs in the frequency domain as a matrix shown in Figure 8.2. In the Frequency Partition methodology, each node program performs beamforming calculations on its own assigned group of frequency bins.

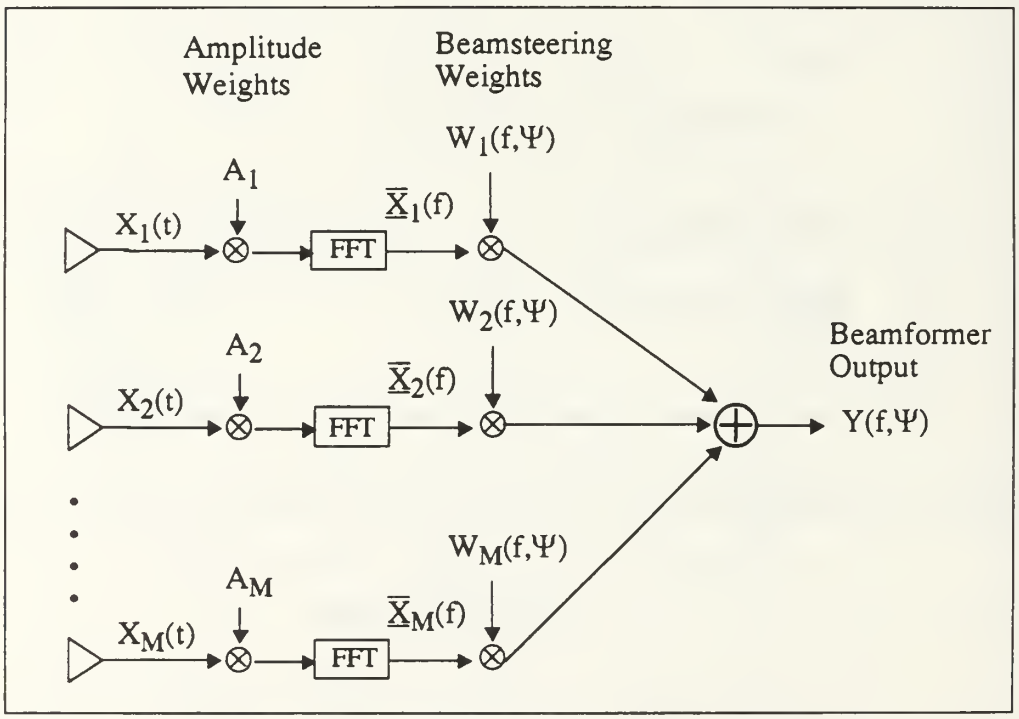

Figure 8.1 Diagram Of Beamformer Algorithm.

Figure 8.2 shows the frequency bins processed by Nodes 0, 1, and 7 in an eight-node parallel system. For a 256 point FFT, only the positive frequency components consisting of 128 points are used for steering the beam. Therefore, to evenly distribute the computational load, each node processes a section of 16 frequency bins from each hydrophone output. The benefit of this partitioning method is a reduction in the amount of data transmitted between nodes before steering. Figure 8.3 shows the overall host and node program structures for a four node parallel system.

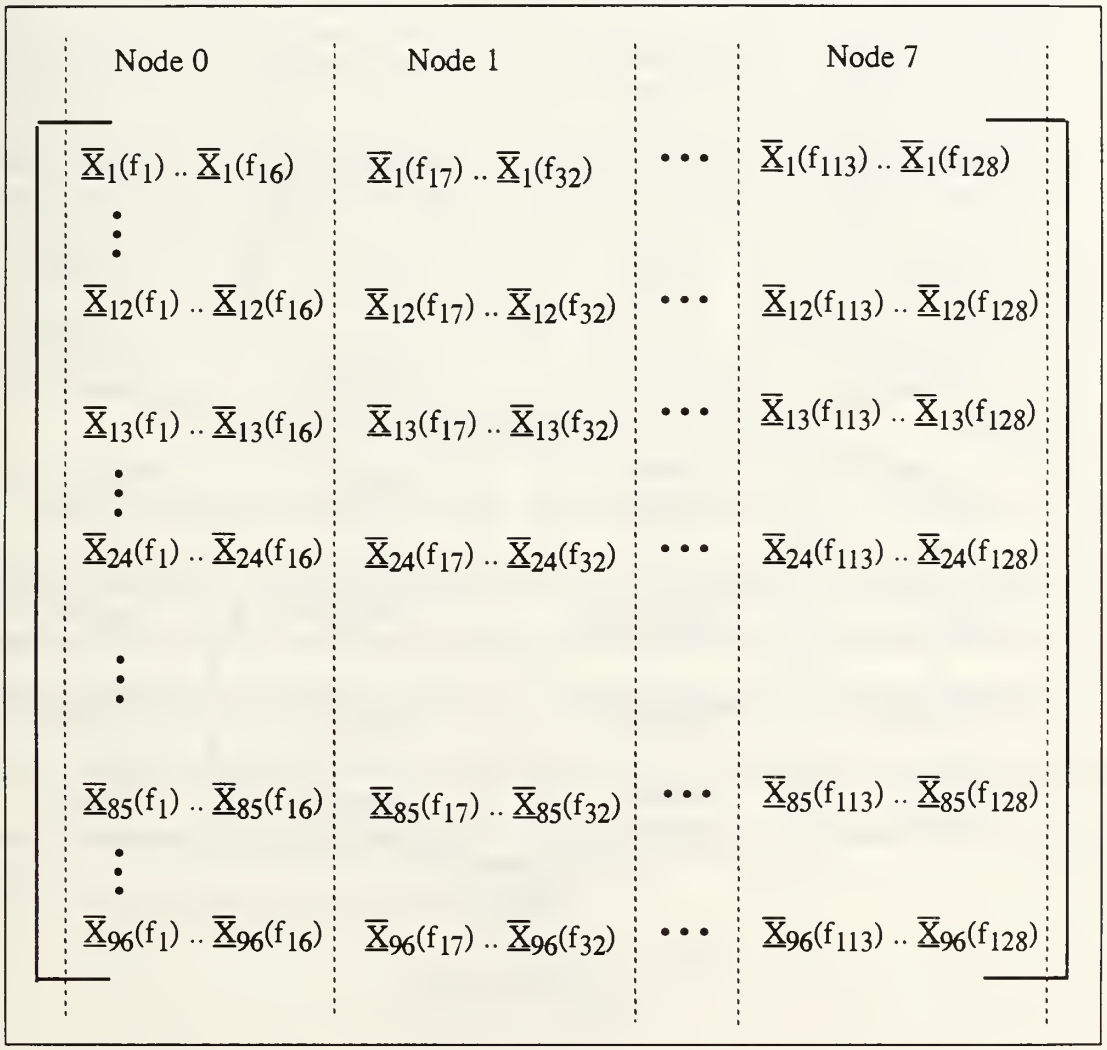

Figure 8.2 Matrix of hydrophone data partitioned by frequency bins.

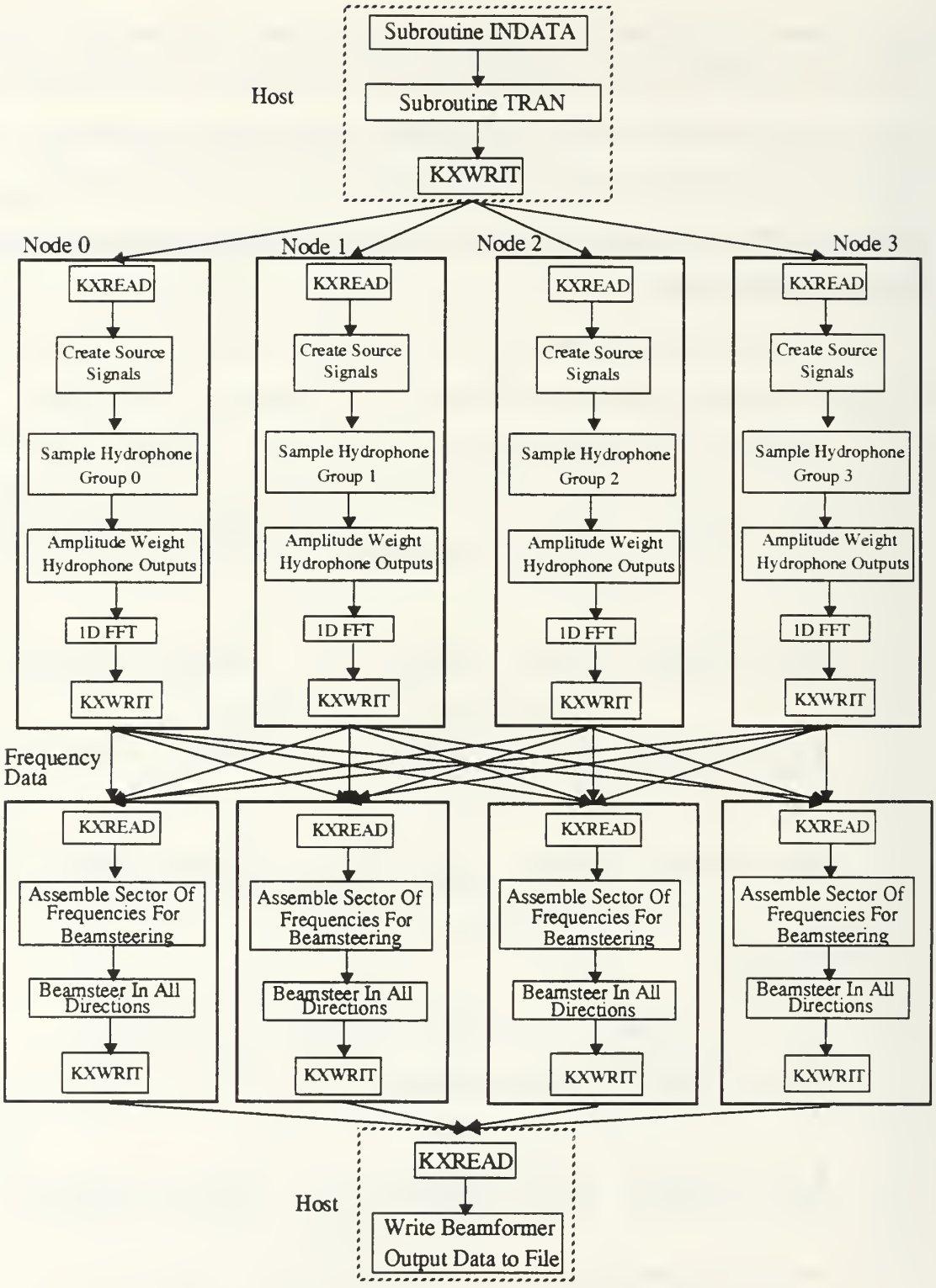

Figure 8.3 Four processor Frequency Partition With Message Shuffle program.

TABLE XIII: PROFILING DATA FOR EIGHT NODE FREQUENCY PARTITION WITH MESSAGE SHUFFLE

Node	Calculation Time (msec)	Node Communication Time (msec)	Input/ Output (msec)	System Calls (msec)	Idle Time (msec)
Node 0	10553.71	1315.81	0.00	0.00	1569.01
Node 1	11143.35	1366.78	0.00	0.00	927.28
Node 2	11177.26	1393.18	0.00	0.00	886.89
Node 3	12370.29	1064.03	0.00	0.00	23.03
Node 4	11232.63	1563.71	0.00	0.00	649.98
Node 5	11261.46	1554.88	0.00	0.00	622.80
Node 6	11239.10	1721.20	0.00	0.00	489.76
Node 7	10195.78	1636.01	0.00	0.00	1621.73
Average	11146.70	1451.95	0.00	0.00	847.93

The Frequency Partition simulation runs were conducted using the same inputs discussed in the previous chapters. Table XIII and Figure 8.4 show the node utilization for an eight-node parallel beamformer. Tables XIV and XV illustrate that increasing the number of processors resulted in speedup. Figures 8.5 and 8.6 show the plots of total processing time and speedup for the different runs. The Frequency Partition method greatly reduced the node communication and idle time compared to the previous partition methods. This is the main reason for the greater speedup.

TABLE XIV: PROFILING DATA FOR FREQUENCY PARTITION WITH MESSAGE SHUFFLE

Number of Processors	Calculation Time (msec)/ Percent Of Total	Node Communication Time (msec)/ Percent Of Total	Input/ Output (msec)/ Percent Of Total	System Calls (msec)/ Percent Of Total	Idle Time (msec)/ Percent Of Total
One	81839.53 99.36%	508.99 0.62%	0.00 0%	0.00 0%	16.76 0.02%
Two	40313.36 94.38%	1711.60 4.01%	0.00 0%	0.00 0%	688.95 1.61%
Four	24805.19 90.41%	1681.11 6.13%	0.00 0%	0.00 0%	948.53 3.46%
Six	14660.10 87.38%	1212.08 7.22%	0.00 0%	0.00 0%	905.34 5.40%
Eight	11146.70 82.90%	1451.95 10.80%	0.00 0%	0.00 0%	847.93 6.30%

TABLE XV: TOTAL PROCESSING TIME AND SPEEDUP FOR FREQUENCY PARTITION WITH MESSAGE SHUFFLE

Number of Processors	Average Total Processing Time (msec)	Speedup
One	82365.28	1.00
Two	42713.91	1.93
Four	27434.83	3.00
Six	16777.52	4.91
Eight	13446.58	6.13

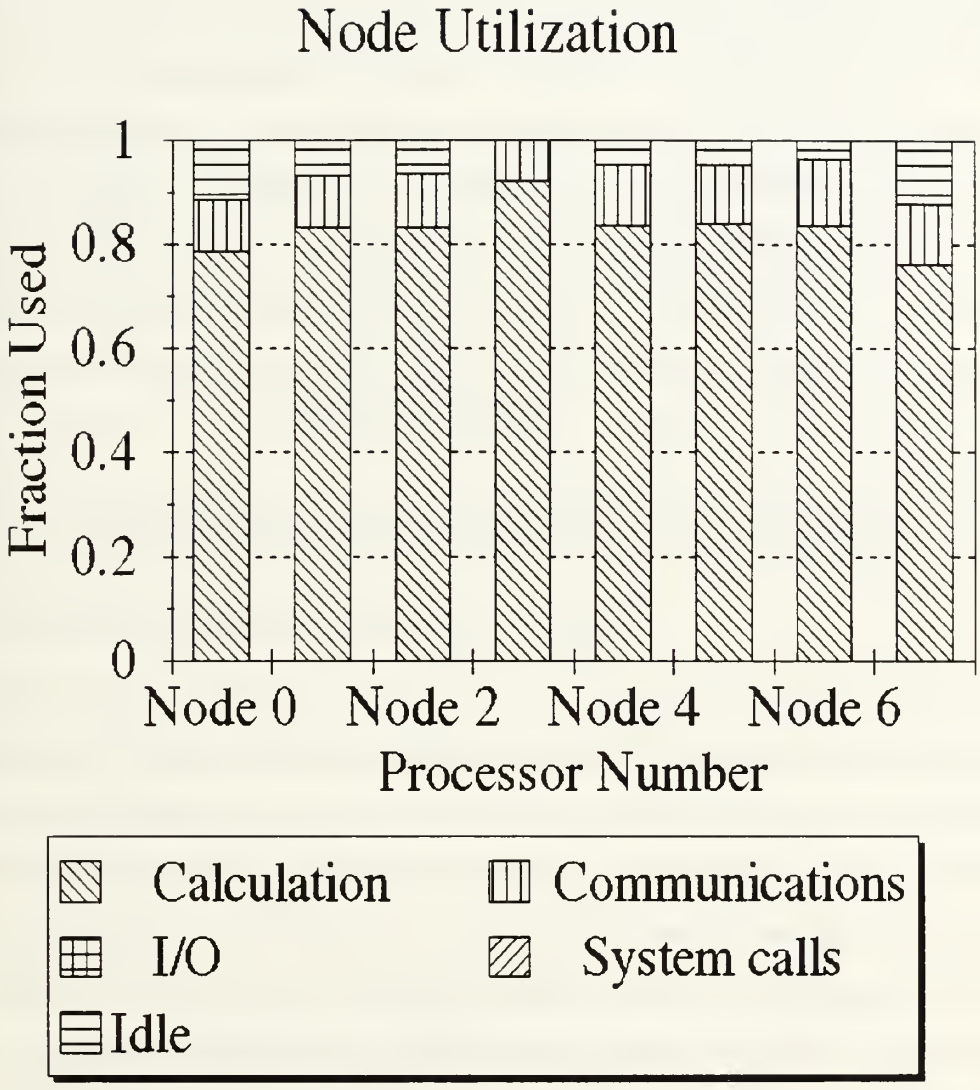

Figure 8.4 Node utilization for eight node Frequency Partition With Message Shuffle.

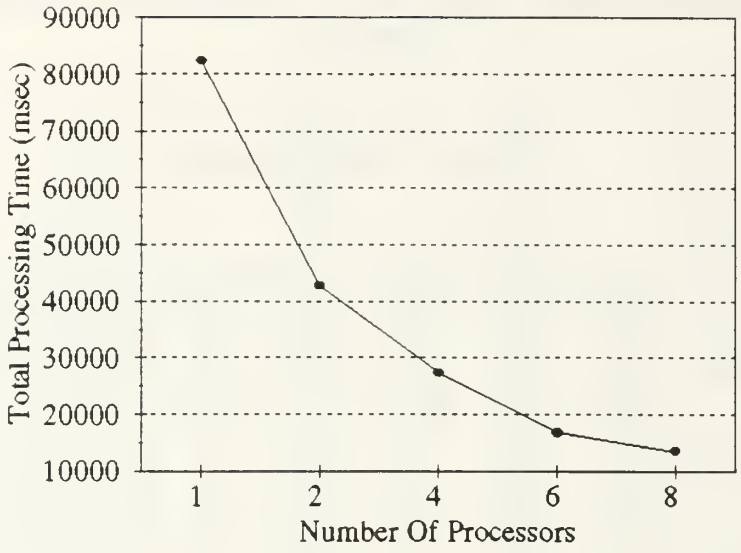

Figure 8.5 Total processing time versus number of processors.

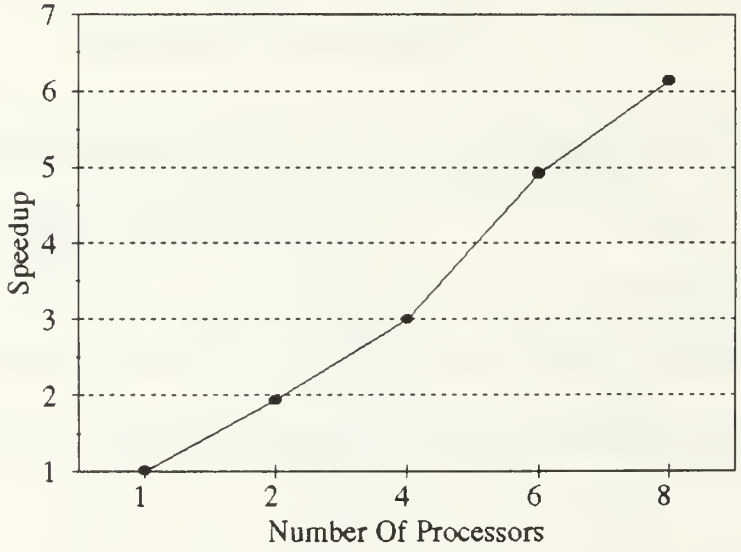

Figure 8.6 Speedup versus number of processors.

IX. TIMING MEASUREMENTS

The performance of a parallel computer system depends on both the computation speed and the communication latencies. Both must be considered when mapping a sequential program to a parallel program. A parallel program may require longer execution time than the sequential program because of slow inter-processor communications. Express provides an event profiling tool called ETOOL to assist in performance analysis [Ref. 4:p. 218]. The programmer places ETOOL markers in a node program. Upon program completion, ETOOL writes a data file listing the time when each marker was encountered during execution. To aid in future development of mapping digital signal processing algorithms, the calculation and communication time of a node program were determined experimentally. All measurements were taken during low network usage times at night.

A. INTER-PROCESSOR COMMUNICATIONS

Table XVI and Figure 9.1 illustrate the inter-processor communication time for transmitting FFT data between processors. The largest FFT length was 2048 points because of the message buffer size limit. The largest number of bytes in one message was experimentally found to be 56,000 bytes. If more than 56,000 bytes are sent in a message, the communications call halts since there is not enough buffer space to store the message.

Figure 9.1 shows a plot of the average communication time. The high and low bars in Fig. 9.1 show the confidence interval. The mean, plus and minus one standard deviation, determines the 68.27% confidence interval [Ref. 8:p. 195]. The data array being transmitted is of the double precision complex type. For comparison, a single integer was sent via the same message function. The average communication time for the integer was 7.79 msec.

A mathematical prediction model was constructed to interpolate and extrapolate average communication times. A second-order polynomial can fit the communication time

data with the least error [Ref. 9:p. 188]. The best fitting polynomial equation for the inter-processor communication time is given by

$$\text{Inter-Processor Communication Time (msec)} = 1.9026\text{x}10^{-5}\,(\text{FFT Length})^2 + 2.6509\text{x}10^{-1}\,(\text{FFT Length}) + 92.56 \quad (9.1)$$

Table XVII compares the actual measured and the interpolated communication time. Figure 9.2 shows the plots of both data sets. The predicted data follow the general shape of the measured data curve. Figure 9.3 shows a plot of the projected inter-processor communication times. These values are presented with the assumption that the processor has enough buffer memory so that the message will not cause blockage.

B. FFT COMPUTATION TIME

The FFT computation time measurements were made in the same program using ETOOL. As with the communication measurement trials, the largest FFT point size was 2048. Table XVIII presents the average computation time along with the standard deviation. The only limitation that may affect the processor FFT computation time is the amount of random access memory (RAM). If there is not enough RAM, part of the data is temporarily written to disk. For all trials, the processor had a total of 16 megabytes of RAM.

Figure 9.4 is a graph of the mean and standard deviation of the FFT computation time. The computation time standard deviations were considerably less than the communication time standard deviations. A polynomial equation was found to fit the FFT computation time data quite well. A second-order equation that best fits the actual results is given by

$$\text{FFT Computation Time (msec)} = 3.5544\text{x}10^{-5}(\text{FFT Length})^2 + 5.027\text{x}10^{-1}(\text{FFT Length}) + 232.51 \quad (9.2)$$

Table XIX compares the actual computation time with the predicted time. Figure 9.5 shows a graph of both data sets. The percent error between the actual and predicted data is in the single digit range. Figure 9.6 shows a plot of predicted computation time with FFT lengths greater than 2048 points. This projection is given with the assumption that the processor has enough RAM to perform the calculations and does not need to temporarily offload data to disk.

In this chapter attempts were made to construct quadratic models that can predict the FFT computational time and the Ethernet communication time. Future work using these models may help to predict the scalability of parallel systems.

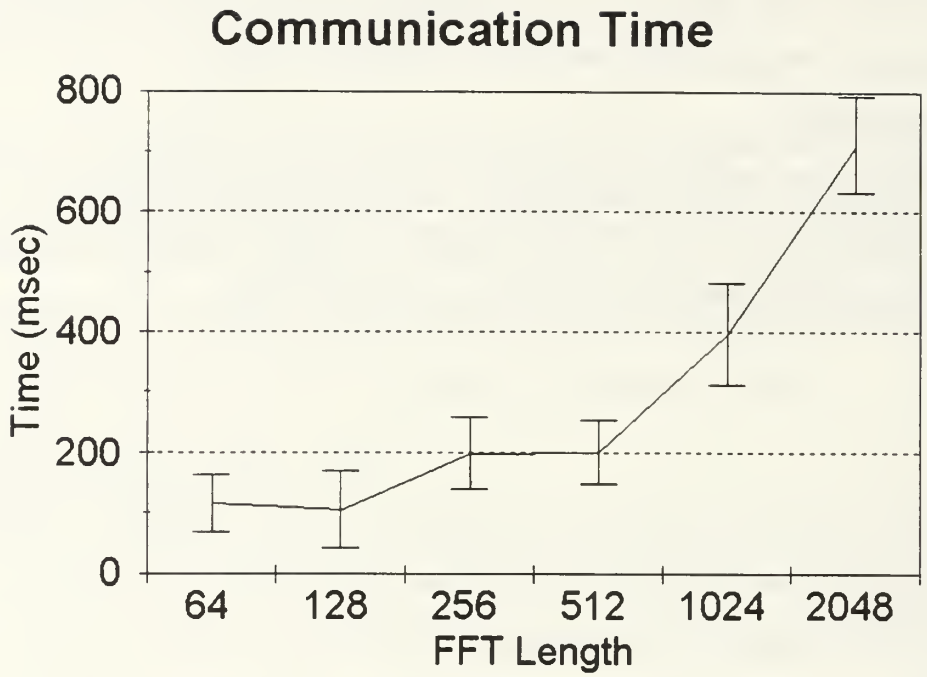

Figure 9.1 Parallel computer network communication time for FFT messages.

TABLE XVI: PARALLEL COMPUTER SYSTEM INTER-PROCESSOR COMMUNICATION TIME

FFT Length	Average Time (msec)	Standard Deviation (msec)
64	115.57	47.91
128	105.64	64.49
256	198.69	59.35
512	200.38	52.19
1024	396.69	84.29
2048	713.66	79.88

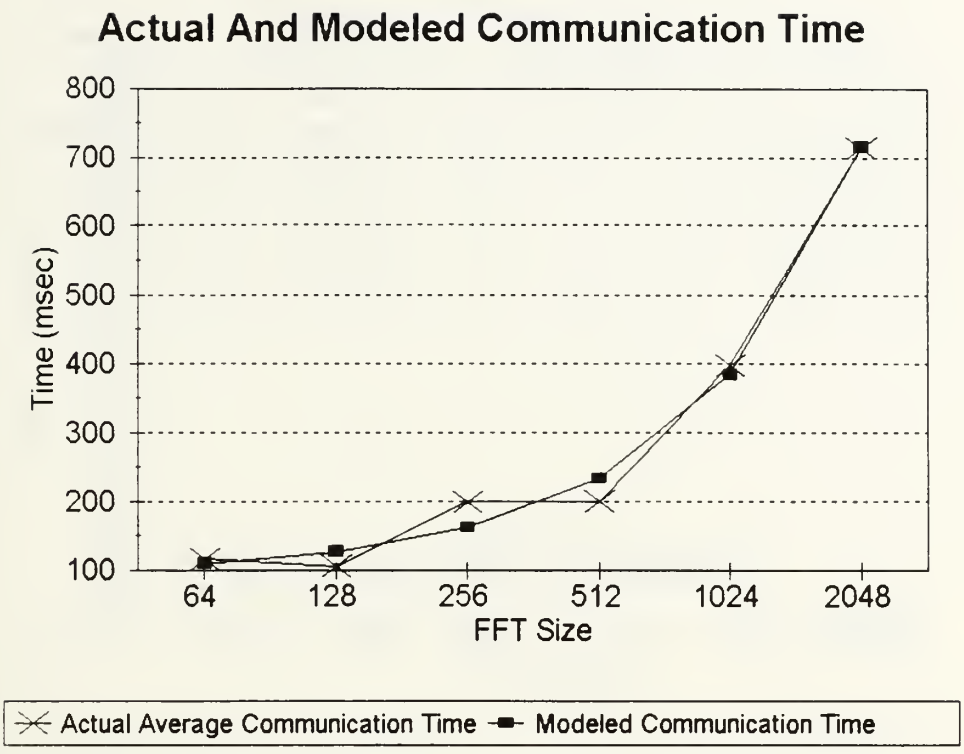

Figure 9.2 Actual and modeled communication time.

TABLE XVII: MODELED INTER-PROCESSOR COMMUNICATION TIME

FFT Length	Actual Average Time (msec)	Modeled Average Time (msec)	Percent Error
64	115.57	109.60	5.16%
128	105.64	126.81	20.04%
256	198.69	161.67	18.63%
512	200.38	233.28	16.42%
1024	396.69	383.97	3.21%
2048	713.66	715.28	0.23%

TABLE XVII: MODELED INTER-PROCESSOR COMMUNICATION TIME

FFT Length	Actual Average Time (msec)	Modeled Average Time (msec)	Percent Error
4096	-	1497.60	-
8192	-	3541.06	-
16384	-	9543.19	-

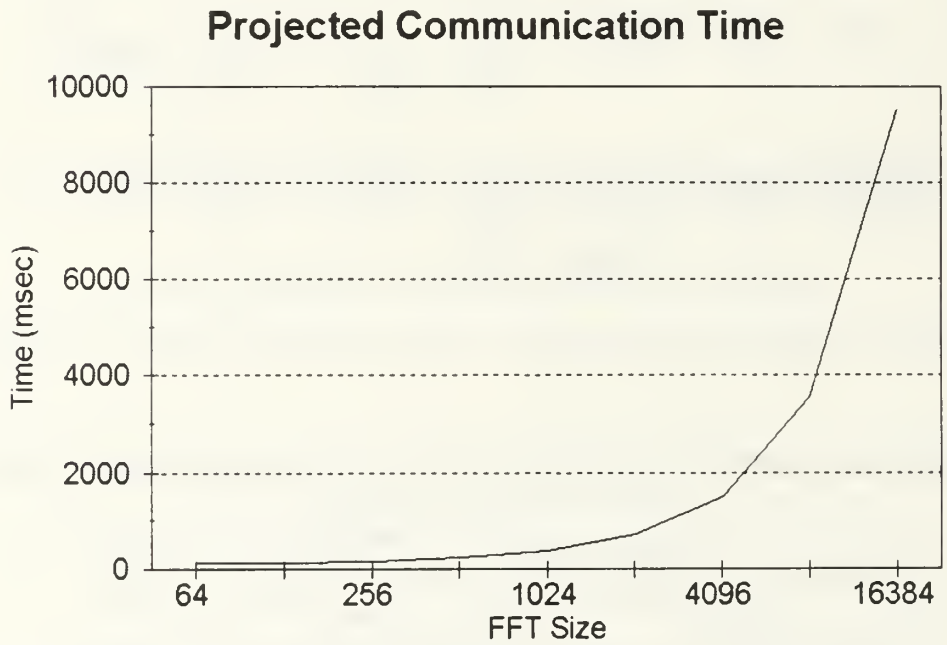

Figure 9.3 Projected communication time for inter-processor communications.

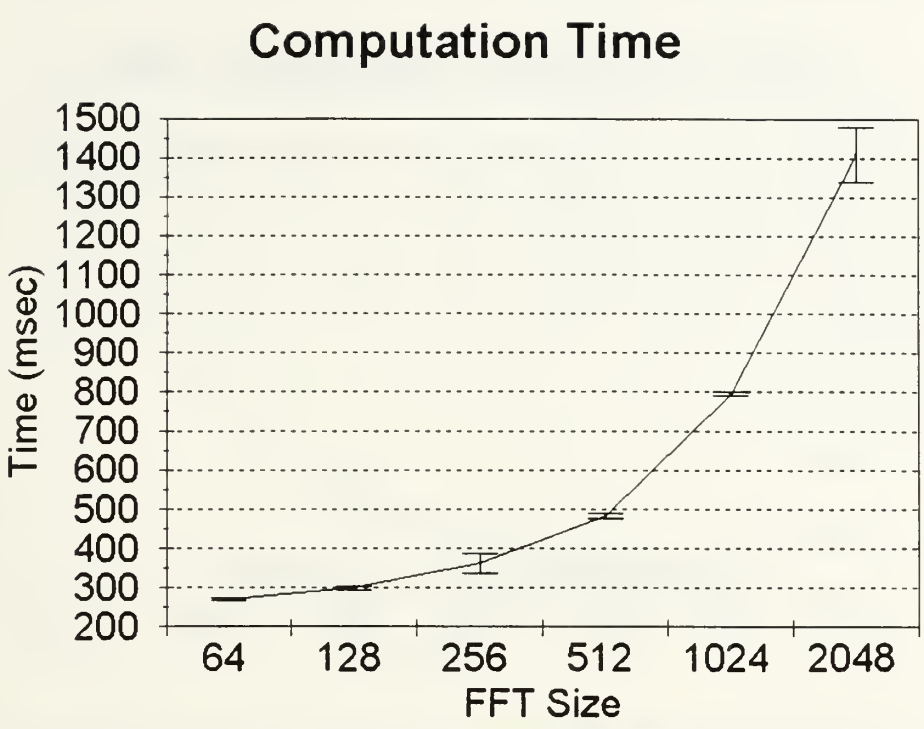

Figure 9.4 Single processor computation time for FFT messages.

TABLE XVIII: PROCESSOR FFT COMPUTATION TIME

FFT Length	Average Time (msec)	Standard Deviation (msec)
64	270.37	3.68
128	299.74	4.08
256	361.97	24.55
512	483.49	5.80
1024	795.81	4.23
2048	1409.30	69.99

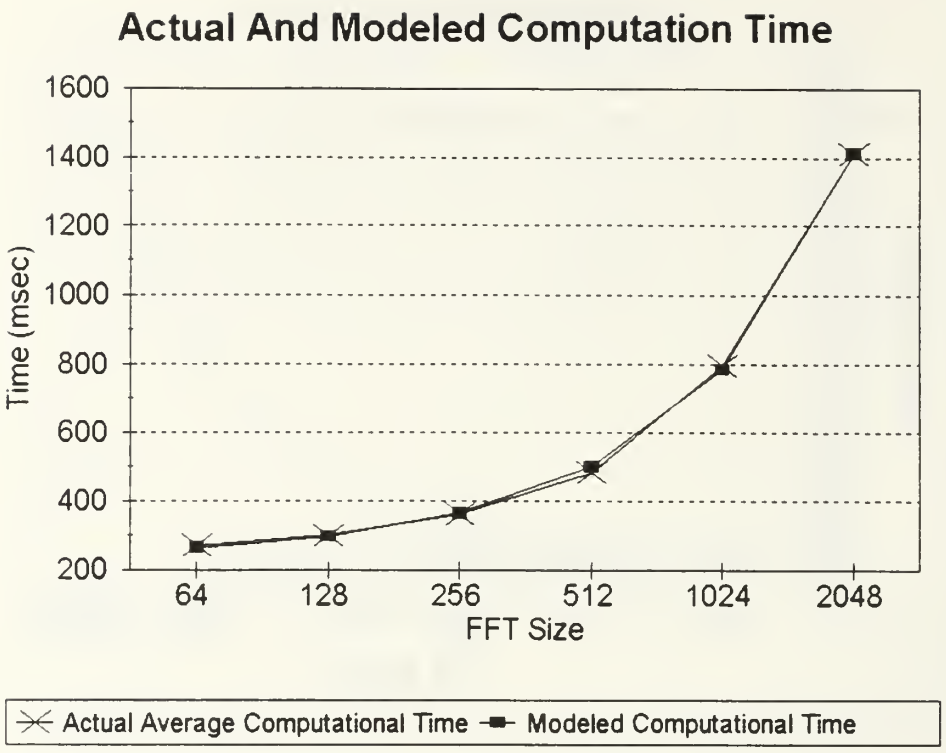

Figure 9.5 Actual and projected processor FFT computation time.

TABLE XIX: MODELED PROCESSOR FFT COMPUTATION TIME

FFT Length	Actual Average Time (msec)	Modeled Average Time (msec)	Percent Error
64	270.37	264.83	2.05%
128	299.74	297.44	0.77%
256	361.97	363.53	0.43%
512	483.49	499.21	3.25%
1024	795.81	784.55	1.41%
2048	1409.30	1411.12	0.13%

TABLE XIX: MODELED PROCESSOR FFT COMPUTATION TIME

FFT Length	Actual Average Time (msec)	Modeled Average Time (msec)	Percent Error
4096	-	2887.97	-
8192	-	6735.96	-
16384	-	18010.07	-

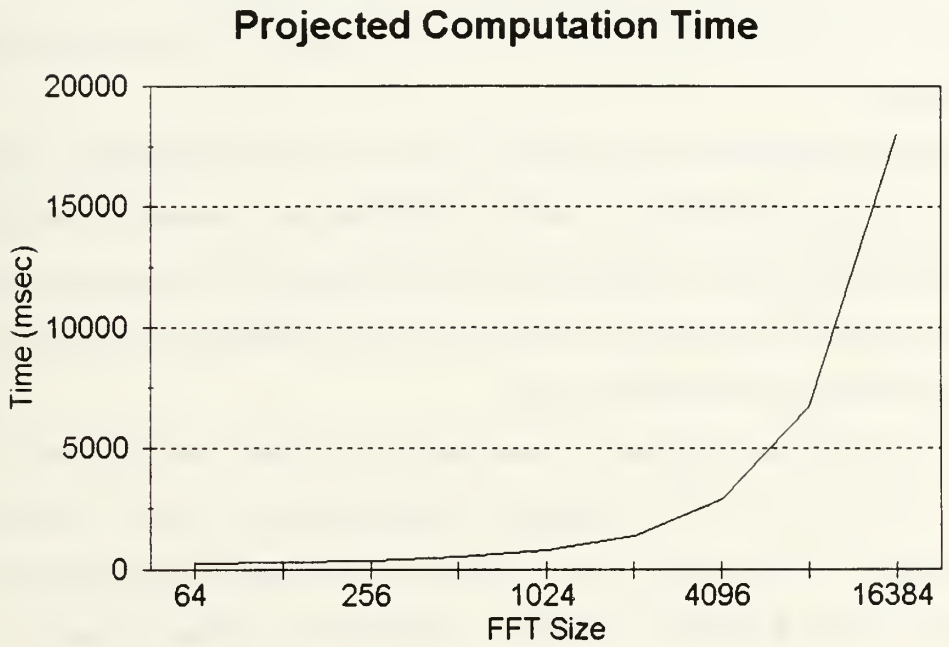

Figure 9.6 Projected processor FFT computation time.

X. CONCLUSION

This thesis research compared two parallel programming methodologies: Loosely Synchronous Communication Partition and Host-Node Partition. Loosely Synchronous programming is the easiest method for converting sequential programs to parallel programs. However, the ease in programming is traded off for lower performance as the number of processors is increased. As shown in Figures 8.1 and 8.2, the Loosely Synchronous Communication Method For Partition experienced longer processing time and a decrease in speedup for more than four processors. With more than four processors, processing time increases because each node makes frequent system call requests to the host program.

Host-Node Programming Methodology offers the best programming paradigm to achieve an increase in speedup as the number of processors are increased. Several partitions were implemented to find the mapping that had the best speedup in the parallel workstation systems. The parallel programming process can best be described as repetitiously trying different partitioning and mapping designs.

For the beamformer application considered in this thesis, two key concepts aided speedup. The first was using broadcast communications in place of point-to-point communications. Broadcasting common data greatly reduced node program idle time resulting in increased speedup. The second concept was to reduce the amount of data transmitted in the network. The success of the Frequency Partition method was due to transmitting less hydrophone data from each node compared to the amount of data transmitted using the other partition methods.

This thesis research can be continued by improving communication and computer capabilities. Since the Ethernet facilitated the inter-processor communications between workstations, a new communications medium can be used. Inter-processor communication time using fiber optic cable can be explored. Besides experimenting with inter-workstation

communications, the beamformer simulation can be run on a supercomputer. A supercomputer offers more nodes and another method of inter-processor communications. Pipelining can be used to increase speedup. Programming with more than eight nodes and using piplelining allow new partitioning schemes to be developed.

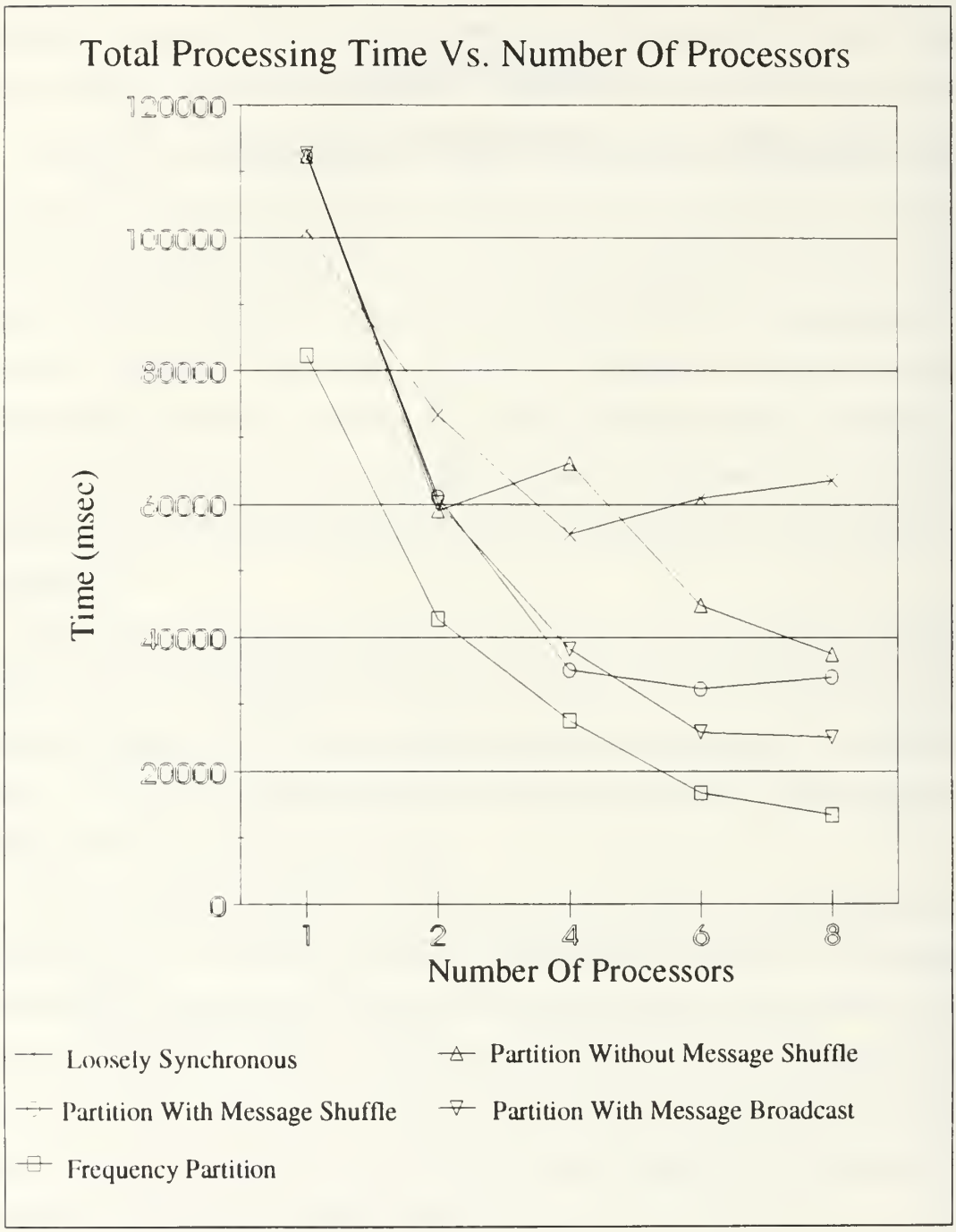

Figure 10.1 Total processing time vs. number of processors for all partition methods.

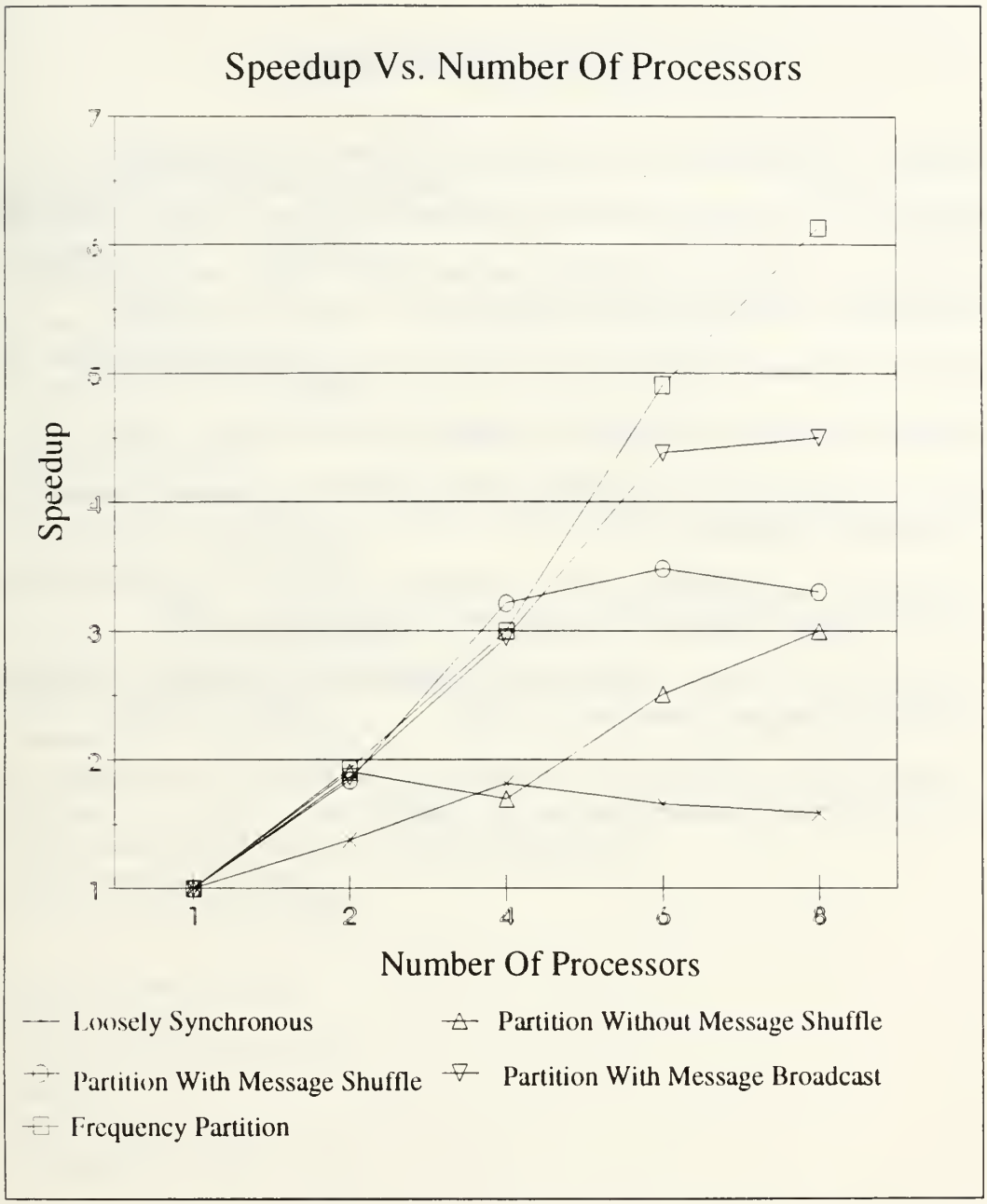

Figure 10.2 Speedup vs. number of processors for all partition methods.

LIST OF REFERENCES

1. Bryan, J., and Pountain, D., "All Systems Go," *Byte*, v. 17, August 1992.
2. Pillai, S. U., *Array Signal Processing*, Springer-Verlag Inc., New York, NY., 1989.
3. Ziomek, L. J., "Fundamentals Of Acoustic Field Theory And Space-Time Signal Processing," Aksen Associates, Homewood, IL., manuscript in progress.
4. Parasoft Corporation, Pasadena, CA., *Express Fortran User's Guide*, 1990.
5. Parasoft Corporation, Pasadena, CA., *Express Fortran Reference Guide*, 1990.
6. Flower, J. and Kolawa, A., "A Packet History Of Message Passing Systems," Parasoft Corporation, Pasadena, CA.
7. Hennessy, J. L., and Patterson, D. A., *Computer Architecture: A Qualitative Approach*, pp. 8-9, Morgan Kaufmann Publishers, Inc., San Mateo, CA., 1990.
8. Spiegel, M. R., *Theory and Problems Of Probability And Statistics*, McGraw-Hill Publishing Company, New York, NY.,1975.
9. Giordano, F.R., and Weir, M.D., *A First Course In Mathematical Modeling*, Brooks/ Cole Publishing Company, Belmont, CA., 1985.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 22304-6145

2. Library, Code 52 — 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Chairman, Code EC — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor C. H. Lee, Code EC/Le — 2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor L. J. Ziomek, Code EC/Zm — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Chief Of Naval Research — 1
   Ballston Tower 1
   Attn: Ms. Elizabeth E. Wald
   Code 227
   800 N. Quincy St.
   Arlington, VA 22217-5660

7. Chief Of Naval Research — 1
   Ballston Tower 1
   Attn: CDR Grace L. Thompson
   Code 227A
   800 N. Quincy St.
   Arlington, VA 22217-5660

8. Mr. Steven L. Howell 1
Naval Surface Warfare Center
Dahlgren Division, White Oak Detachment
Silver Spring, MD 20903-5000

9. LT Daniel T. Sullivan, USN 2
219 Sherman Ave.
Montgomery, IL 60538

Thesis
S8585 Sullivan
c.1 Computer simulation studies of two-dimensional beamforming for linear arrays using a parallel computer system.